DESARROLLO DE PRACTICAS DE LABORATORIO CON MICROCONTROLADORES PARA DOCENCIA UNIVERSITARIA

Hernan Alonso Bravo Vaquero

Humberto Cabra Tamayo

CORPORACIÓN UNIVERSITARIA AUTÓNOMA DE OCCIDENTE

DIVISIÓN DE INGENIERÍAS

PROGRAMA DE INGENIERÍA ELÉCTRICA

SANTIAGO DE CALI

1996

DESARROLLO DE PRACTICAS DE LABORATORIO CON MICROCONTROLADORES PARA DOCENCIA UNIVERSITARIA

Hernan Alonso Bravo Vaquero

Humberto Cabra Tamayo

Trabajo de grado para optar al título de
Ingenieros Electricistas

Director

Fabio Almanzar Gonzalez
Ingeniero Electrónico

CORPORACIÓN UNIVERSITARIA AUTÓNOMA DE OCCIDENTE

DIVISIÓN DE INGENIERÍAS

PROGRAMA DE INGENIERÍA ELÉCTRICA

SANTIAGO DE CALI

1996

NOTA DE ACEPTACIÓN

Aprobado por el comité de grado en cumplimiento de los requerimientos exigidos por la Corporación Universitaria Autónoma de Occidente para optar al título de Ingeniero Electricista.

DIRECTOR

JURADO

JURADO

Santiago de Cali, Agosto de 1.996

AGRADECIMIENTOS

Al hacer entrega de este proyecto a la comunidad académica, los autores expresan sus agradecimientos:

A todo el grupo de docentes que nos brindaron la formación de Ingenieros Electricistas durante el transcurso de nuestra vida universitaria.

A **Fabio Almánzar González**, vinculado a la Corporación Universitaria Autónoma de Occidente como profesor hora cátedra del Programa Ingeniería Electrónica y Director de la tesis, por sus invaluables aportes en todas las etapas desarrolladas en este proyecto.

A **Oscar Fernando Agredo Satizabal**, vinculado a la Corporación Universitaria Autónoma de Occidente como Director del Programa Ingeniería Electrónica y Asesor de la tesis, por sus aportes y confianza depositada en nosotros.

A **Henry Cabra Tamayo**, vinculado a la Corporación Universitaria Autónoma de Occidente como profesor tiempo completo del programa de Ingeniería Electrónica por los aportes y apoyo ofrecidos, los cuales fueron de gran ayuda para la realización de este proyecto.

A **Zeida María Zolarte,** vinculada a la Corporación Universitaria Autónoma de Occidente como directora del área de Digitales del programa de Ingeniería Electrónica por la colaboración en la redacción del libro guía de este proyecto.

A **Javier Parra,** vinculado a la Corporación Universitaria Autónoma de Occidente como profesor hora cátedra, por la colaboración en la redacción del libro guía de este proyecto.

A todos los profesores de Ingeniería Electrónica que de alguna manera colaboraron en la culminación de este proyecto.

DEDICATORIA

Al culminar otra etapa de mi vida llena de sacrificios y grandes satisfacciones, quiero dedicar un especial reconocimiento a mis padres, **María Cecilia** y **Luis Humberto,** por su esfuerzo y constante apoyo en el desarrollo y terminación de mis estudios, quienes me enseñaron a enfrentar con decisión todos los compromisos y retos que se adquieren a lo largo de la vida; a mis hermanos, **Nancy, Freddy, Henry, Cecilia**, por su preocupación y constante apoyo; a mi novia, **Claudia Patricia Naranjo**, por la confianza que me ha brindado y su constante apoyo y respeto; a mis familiares quienes creyeron y aguardaron pacientes la culminación de esta meta y a DIOS por iluminarme y guiarme en todo momento.

Humberto Cabra Tamayo

DEDICATORIA

Al culminar una etapa más en mi vida y una de las más importantes, quiero dedicar este logro especialmente a la memoria de mi madre **Cecilia Baquero** la cual se fue de esta vida terrenal con la esperanza de ver culminar mis estudios universitarios y a su vez triunfar en mi vida profesional, a mi padre **Ezequias Bravo** quien en compañía de mi madre me brindaron su esfuerzo y constante apoyo en el desarrollo y terminación de mis estudios, quienes me enseñaron a enfrentar con decisión todos los compromisos y retos que se adquieren a lo largo de la vida; a mis hermanos **Luis, Ruby, Blanca, Aura, Oscar, Guido y Harold**, y en general a toda mi familia, los cuales me apoyaron y motivaron en todo instante a seguir adelante; a **Regina Betancur de Liska (Regina 11)** quien con sus enseñanzas de Metafisica me enseñó a ver la vida desde otro punto de vista y a que todo lo que uno se propone en la vida lo consigue y a DIOS por iluminarme y guiarme en todo momento

Hernan Alonso Bravo Vaquero

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

Este proyecto esta dividido en dos partes: En la primera se ha recopilado una serie de información acerca de los microcontroladores, de tal manera que se convierta en un libro de consulta permanente y, además, sirva como punto de partida o referencia para seleccionar el tipo de microcontrolador más acorde a la necesidad planteada.

En la segunda parte, se implementan cuatro prácticas de laboratorio que ayudan a comprender el funcionamiento general de cualquier sistema de microcontroladores y aún de microprocesadores.

Esta organizado siguiendo un orden lógico de aprendizaje y utilización del sistema de desarrollo **Microlab-51**. Inicia con un tema básico y esencial, introducción a los microprocesadores, el cual es indispensable para un mejor entendimiento de los microcontroladores. En el capítulo 2 se presentan las diferentes familias de microcontroladores que existen en el mercado, herramientas de desarrollo, entre otros. El tercer capítulo es el más importante, ya que en el está recopilada la información del microcontrolador 8051 y finaliza con las prácticas de laboratorio implementadas y sugeridas.

Además incluye un disquete con un software para comunicaciones llamado **MICROWIN**. Finalmente en un anexo se incluye el manual de usuario del software.

0. INTRODUCCIÓN

Uno de los campos menos explorados en el país tanto a nivel científico como económico, es el apoyo que pueda prestar el microprocesador o microcontrolador en la enseñanza y específicamente en el campo de los laboratorios, donde existe una deficiencia en cuanto a análisis y desarrollo de pruebas en forma confiable.

Una de las áreas con más desarrollo en la actualidad en el campo de la electrónica es la Digital, específicamente aplicaciones de microprocesadores o microcontroladores en procesos o controles automáticos, que poco a poco, han ido obligando a la industria a modernizar y automatizar sus equipos con el fin de tener un rendimiento mucho más alto en sus líneas de producción. El gran desarrollo de la microelectrónica conlleva a la elaboración de circuitos integrados más pequeños, versátiles, de gran velocidad y con funciones cada vez más complejas. Es así como se ha pasado del circuito básico de switcheo en semiconductores como lo es el transistor, a circuitos integrados digitales de baja escala (LSI); posteriormente, se agruparon varias funciones de LSI y dieron origen a la integración a media escala (MSI); actualmente, se trabaja con tecnología

LSI e integración a grande y muy grande escala (VLSI) lo que implica circuitos integrados altamente desarrollados en funciones (programables en un lenguaje propio) y en arquitectura, ya que poseen una unidad central de proceso, memoria propia, unidad de control, unidades de entrada y salida (I/O), todo en un solo chip.

Con este proyecto se entregaran las normas básicas que deben seguirse para diseñar una aplicación específica, buscando impulsar el desarrollo de aplicaciones más complejas que actualicen las tecnologías electrónicas existentes en nuestro medio.

Este proyecto esta dividido en dos partes: En la primera se ha recopilado una serie de información acerca de los microcontroladores, de tal manera que se convierta en un libro de consulta permanente y, además, sirva como punto de partida o referencia para seleccionar el tipo de microcontrolador más acorde a la necesidad planteada.

Los factores que influyen en la utilización de un microcontrolador en particular, son: la permanente disponibilidad en el mercado, su costo, la información que sobre éste se tenga, sus características técnicas y la aceptación por parte de la industria en el diseño de aplicaciones con este tipo de dispositivos. Por lo anterior se recomienda utilizar el microcontrolador de la familia MCS51 de la compañía INTEL para desarrollar las prácticas de laboratorio.

En la segunda parte, se implementan cuatro prácticas de laboratorio que ayudan a

comprender el funcionamiento general de cualquier sistema de microcontroladores y aún de microprocesadores.

Este proyecto tiene la característica de poder ser utilizado por cualquier persona con algún conocimiento en electrónica digital, por estudiantes que realicen un proyecto de grado basados en este sistema de desarrollo, como también por profesionales relacionados con la electrónica industrial, sirviéndoles como una fuente de información que los oriente en la solución de sus diseños.

Este libro esta organizado siguiendo un orden lógico de aprendizaje y utilización del sistema de desarrollo **Microlab-51**. Inicia con un tema básico y esencial, introducción a los microprocesadores, el cual es indispensable para un mejor entendimiento de los microcontroladores y finaliza con las prácticas de laboratorio implementadas y sugeridas.

La distribución de contenidos se realiza de la siguiente forma:

La primera parte: contiene los capítulos 1, 2 y 3.

- En el capítulo 1 se hace una introducción a los microprocesadores, se presenta una reseña histórica para mostrar como han evolucionado, desde la aparición del 4004 de la compañía INTEL, pasando por desarrollos de otras compañías como Zilog,

Motorola, etc., hasta el PENTIUM, además se muestra un diagrama de bloques explicando las partes que lo conforman.

- En el capítulo 2 se presentan las diferentes familias de microcontroladores que existen en el mercado, sus principales características y las ventajas que ofrecen unos con respecto a otros; también se hace referencia a las herramientas de desarrollo con que cuentan los microcontroladores, que permiten desarrollar y depurar sus programas; este capitulo concluye con una serie de recomendaciones al momento de seleccionar un microcontrolador.

- El tercer capítulo es el más importante, ya que en el está recopilada la información del microcontrolador 8051: se explica brevemente el hardware, los modos de direccionamiento y las instrucciones; para dar una idea general del microcontrolador. También se hace referencia a los fabricantes que han basado sus productos en el chip 8051, los cuales han contribuido a que proliferen las más diversas soluciones integradas alrededor del corazón del 8051.

Segunda Parte: Está conformada por el capítulo 4. Esta sección contiene cuatro prácticas de laboratorio las cuales incluyen aplicaciones con diagramas de flujo y sus respectivos programas, como también sugerencias para complementarlos. Con esto se abarca lo correspondiente al manejo del set de instrucciones, manejo de memoria y

modos de comunicación de este microcontrolador.

Todas las prácticas han sido realizadas con el sistema de desarrollo **Microlab-51**, el cual puede ser conectado a un puerto serie de un computador y controlado por el programa de comunicaciones Microlab Editor (MLED), o por el programa de comunicaciones MICROWIN, que además de tener las características del MLED, cuenta con ensambladores y simuladores de la compañía AVOCET para microcontroladores de la familia 80C51 y microprocesadores Z80, también, permite recibir directamente por pantalla la información transmitida por cualquier sistema de desarrollo.

Además hemos adicionado un anexo el cual contiene el manual de usuario del software de comunicaciones que realizamos: **MICROWIN**

1.0 MICROPROCESADORES

1.1. INTRODUCCIÓN

1.1.1. Evolución Histórica

En 1971 la Compañía Intel anunció la aparición del primer microprocesador denominado el 4004. Éste era de 4 bits, estaba implantado con tecnología **PMOS**[1]. Tenía 45 instrucciones y ejecutaba 60.000 operaciones por segundo.

Al siguiente año, la misma Compañía introdujo el 8008, el primer microprocesador de 8 bits; también estaba implantado con tecnología **PMOS**. El primer 8008, además de tener una longitud de palabra mayor, contaba con 48 instrucciones, podía ejecutar 300.000 operaciones por segundo y direccionaba 16 Kbytes de memoria; sin embargo, para poder funcionar requería de aproximadamente 20 circuitos de soporte. Hasta ese

[1] PMOS: P-channel Metal-Oxide Semiconductor

momento el principal objetivo de los microprocesadores era reemplazar compuertas pequeña escala de integración (**SSI**) y Mediana escala de integración (**MSI**).

Avances posteriores en la tecnología de circuitos integrados permitieron que a principios de 1974 Intel anunciara el 8080, un microprocesador de 8 bits mucho más poderoso. El 8080 tenía 78 instrucciones en las cuales se incluían todas las del 8008; además su velocidad era de 10 veces mayor que la del 8008 y podía direccionar hasta 64 Kbytes de memoria. Su tecnología de fabricación fue **N**mos (mayor densidad de integración) lo cual redujo notoriamente los circuitos de soporte.

Sobretodo la principal diferencia del 8080 con respecto a los microprocesadores anteriores, era que no había sido diseñado simplemente para sustituir compuertas lógicas, sino, que podía realizar todas las operaciones de una computadora. Esto originó una revolución tecnológica que se ha venido desarrollando hasta nuestros días.

El 8080 fue un microprocesador de los más populares, convirtiéndose en un estándar de la industria. Dicho microprocesador requería de 3 fuentes y varios chips auxiliares, además junto con el 8080 se introdujeron algunos circuitos periféricos como el 8255 **PPI**[2], entre otros de su mismo género.

En respuesta al éxito del 8080, la Compañía Motorola introdujo, en 1974, un

---

[2] PPI: Periferical Programable Interface

microprocesador de 8 bits, con 72 instrucciones: el 6800. Al mismo tiempo, apareció una familia de circuitos periféricos, diseñados especialmente para conectarse a este microprocesador.

En 1975, la Compañía Mos Technologic anunció 2 microprocesadores, el 6501 compatible con el 6800 y el 6502 el cual incluía un 6501 y además un generador de señal de reloj.

En 1976, la Compañía Zilog introdujo el Z80, un microprocesador Nmos de 8 bits, basado en el 8080 pero apreciablemente mejorado. El Z80 resultó ser un microprocesador mucho más rápido y fácil de usar, podía ejecutar las 78 instrucciones del 8080, así como 80 instrucciones más.

Junto con el Z80, Zilog introdujo varios circuitos periféricos tales como: el controlador de puertos en paralelo, Z80 PIO; el controlador de puertos en serie, Z80 SIO y un circuito timer contador, Z80 CTC.

En 1977, Intel anunció el microprocesador 8085, al igual que el Z80 estaba fabricado con tecnología Nmos y requería un único voltaje de 5V. Junto al 8085 se crearon dos circuitos periféricos especiales para poder formar un sistema microprocesado completo; uno de ellos con memoria RAM, puerto de entrada/salida y un timer; el 8155 y el otro con memoria ROM y puertos, el 8355 ó el 8755.

En 1978 Intel introdujo al mercado el 8086 (de 16 bits) siendo este microprocesador el antecesor de los computadores actuales, como el 80386 y 80486.

En la figura 1.1 se muestra el desarrollo cronológico de los microprocesadores

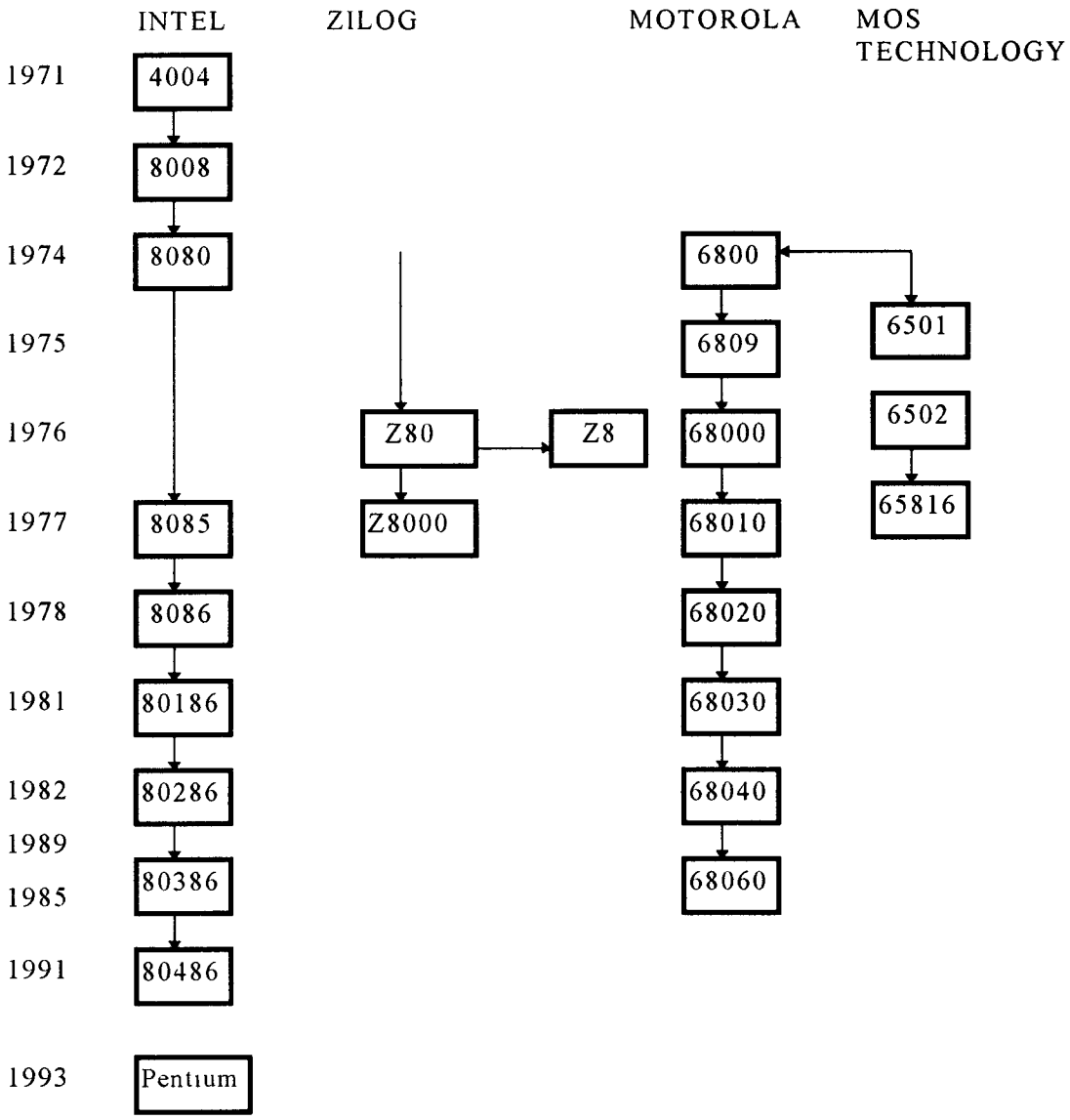


Figura 1.1. Desarrollo Cronológico de los Microprocesadores

1.2. DEFINICIÓN

Un microprocesador es un circuito de alta integración (VLSI), compuesto por la CPU de una computadora sin memoria, entradas/salidas (I/O), muchos circuitos más simples como flip-flops, registros, decodificadores, contadores y los periféricos necesarios para formar un sistema completo. Ejemplos de microprocesadores tenemos los chips 8088, 80286, entre otros. Al combinar entradas/salidas (I/O) y funciones periféricas de memoria podemos formar un microcomputador

Las capacidades de los microprocesadores son muy variadas y con un crecimiento rápido. Se les utiliza en gran variedad de aplicaciones, desde las microcomputadoras de 4 bits de un solo circuito integrado (**CI**), para aplicaciones de gran volumen y bajo costo (juegos electrónicos y los hornos microondas) hasta la generación de procesadores de 16 bits como la Motorola 68000 y el Zilog Z-8000, con capacidades de programación similares a las de una computadora.

El microprocesador es el que se encarga de buscar e identificar instrucciones, decodificar e interpretar su código binario y ejecutarlas. Es capaz de referenciar y seleccionar posiciones de memoria y módulos de entrada/salida, reconoce y responde a ciertas señales de control procedentes del exterior, como las demandas de interrupción o cesión de los buses.

El microprocesador precisa las puertas de salida para comunicar al exterior los resultados del procesamiento. Dispositivos de salida pueden ser pantallas gráficas, impresoras, etc

1.3. ARQUITECTURA DE LOS MICROPROCESADORES

Un microprocesador consta básicamente de las siguientes partes:

1.3.1. Unidad de control: por medio de señales de reloj, la unidad de control mantiene la secuencia de eventos para llevar a cabo una tarea especifica. Esta unidad tiene la capacidad de recibir señales externas interrumpiendo temporalmente su funcionamiento para atenderlas.

1.3.2. Unidad Lógica Aritmética (ALU) la ALU lleva a cabo todas las operaciones aritméticas y lógicas. Entre las principales funciones de la ALU están:

- Suma aritmética.
- Funciones lógicas And, Or y Xor.
- Rotaciones y desplazamientos bidireccionales de datos.

1.3.3. Registros internos: son unidades de almacenamiento temporal. Los hay de uso general y de uso específico. Uno de los registros principales, es el contador de

programa o PC, el cual, almacena la dirección de la próxima instrucción a ejecutar. Otro registro importante es el registro de instrucciones en el cual se almacena la instrucción a ser decodificada. El acumulador es otro registro interno, el cual se utiliza para realizar las operaciones de la ALU.

1.3.4. La Interfaz de Bus: encargada de administrar todos los buses del microprocesador: datos, direcciones y control.

La organización interna, o arquitectura, de la mayoría de los microprocesadores se presenta en la figura 1.2. Cada fabricante aporta a dicha estructura ciertas particularidades, que diferencian sus modelos de los de la competencia, y que proporcionan una mejor acomodación a los requerimientos exigidos por los clientes.

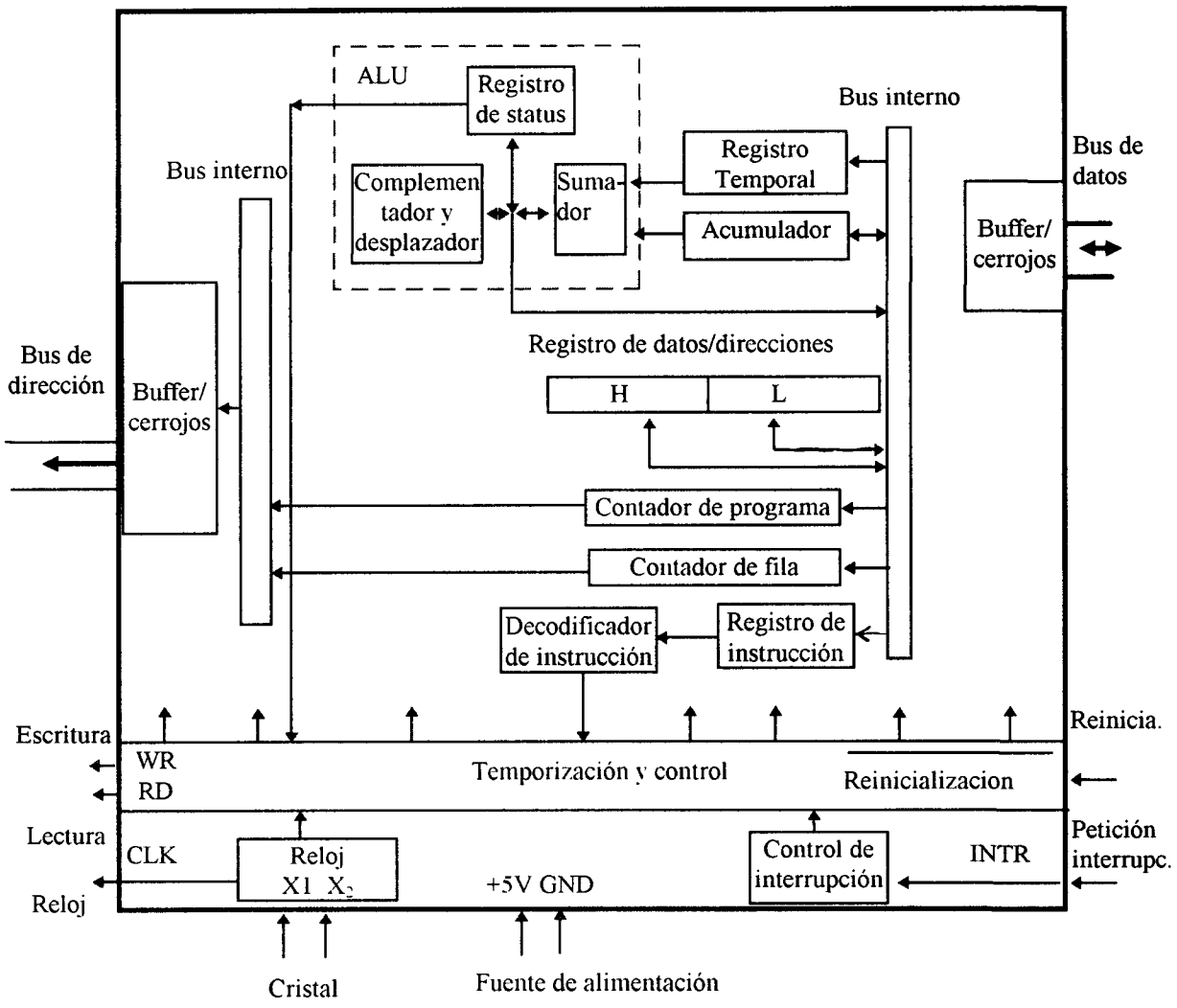


Figura 1.2. Diagrama de bloques funcionales del microprocesador genérico

2.0 MICROCONTROLADORES

2.1. INTRODUCCIÓN

Al final de la década de los 70 aparecen los microcontroladores (μC). Los cuales, surgen como un paso más en el desarrollo de los microprocesadores y para cubrir las necesidades de la industria, en las cuales se requerían sistemas de control de alto rendimiento y confiabilidad. Los microcontroladores fueron posibles, gracias al desarrollo de nuevas técnicas en la fabricación de integrados y mejores escalas de integración.

Los fabricantes de microcontroladores han desarrollado sus productos basados en las versiones de sus microprocesadores, algunos realizaron sus diseños manteniendo compatibilidad con el microprocesador base; otros, cambiaron el concepto al desarrollar su microcontrolador, creándose 2 tendencias: la primera de ellas, incluye aquellos fabricantes que desarrollan un microcontrolador que incluye todo el hardware necesario para una aplicación y que conserva compatibilidad con sus productos

anteriores (microprocesadores) en el caso de la Motorola con su familia de microcontroladores 68HXX , basados en el microprocesador 6800.

La otra tendencia, fue desarrollar un microcontrolador el cual cumpliere con sus requerimientos básicos de hardware, pero, con un concepto diferente en su programación, no guardando compatibilidad con los microprocesadores anteriores, creando sistemas totalmente nuevos, como la Zilog, la cual desarrollo la familia Z8 sin guardar compatibilidad con su microprocesador Z80.

Esta división crea dos filosofías en el diseño de microcontroladores: los que simplifican el "hardware" al incluir muchas partes de él en el chip y los que simplifican el "software" al poseer una estructura de programación diferente.

Entre las principales características de un microcontrolador están:

- Ejecución rápida de instrucciones.
- Uso eficiente de la memoria.
- Líneas de I/O programables.
- Manejo sofisticado de interrupciones.

En la figura 2.1 se presenta el diagrama de un hipotético circuito microcontrolador con la mayoría de los elementos que se utilizan en aplicaciones de medición,

instrumentación y control; este microcontrolador integra los siguientes subsistemas, aunque no siempre presenta todos los relacionados:

- CPU.
- RAM.
- ROM.
- ROM O EPROM.
- Entradas/Salidas.
- Contadores y temporizadores.
- Conversores Análogo/Digital y Digital/Análogo.
- Gestión de interrupciones

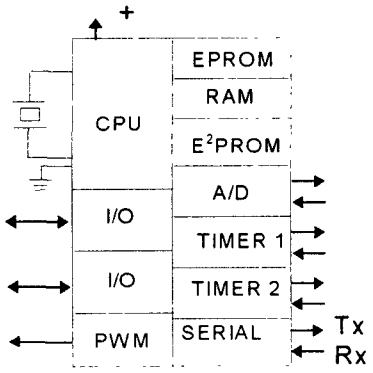


Figura 2.1. Diagrama de un Microcontrolador Ideal

Se puede decir que cuando los sistemas basados en el microprocesador se especializan en aplicaciones industriales, en ambientes eléctricos adversos, aparece la versión industrial del microcomputador monopastilla, que no es otro que, el **microcontrolador**.

Un microcontrolador es un sistema integrado muy atractivo por su aplicación en muchos campos. Las aplicaciones de los microcontroladores, en general, pueden ser subdivididas en dos amplias clases:

- Aplicaciones de control de estado.(Microcontroladores de 8 bits)
- Aplicaciones de control de lazo cerrado.(Microcontroladores de 16 bits)

Los microcontroladores de 8 bits son utilizados en aplicaciones de mediana complejidad, por su facilidad de manejo y nivel de prestaciones. Puede ser utilizado como microprocesador implementando la memoria RAM, EPROM y/o unidad de Entrada/Salida que se considere más adecuada a su aplicación.

Actualmente los microcontroladores de 8 bits se utilizan para la solución de problemas planteados de control en diversos campos como:

•Informática: controlador de periféricos; como control de impresoras, plotters, terminales, disco duros, teclados, modems, etc.

•Industria del automóvil: para controlar el motor, regulación del servomotor, ordenador de ruta, dosificador, alarmas, etc

•Industria de los electrodomésticos: control de calefacción, lavadoras, lavavajillas, cocinas eléctricas, etc.

•Medicina, aplicaciones militares, burótica (máquinas de escribir), edificios inteligentes, etc.

•Industria de imagen y sonido: Tratamiento de la imagen y sonido, control de los motores de arrastre del giradiscos, magnetófono, vídeo, etc.

No obstante, el microcontrolador de 8 bits tiene unas limitaciones de capacidad para afrontar con éxito algunas aplicaciones que exigen unas características de velocidad y precisión críticas.

Las aplicaciones de **control de estado**, presentan problemas complejos. Las salidas del microcontrolador dependen de los estado del sistema y de las transiciones que se producen.

Ejemplos de aplicaciones de este tipo son:

- Controladores de teclado
- Luces semafóricas
- Maniobras de aparatos elevadores, de máquinas herramientas, etc.

-Regulación: de niveles, temperaturas, velocidad de máquinas, etc.

-Automatismo: control secuencial de procesos, por ejemplo, control de máquinas herramientas, apertura y cierre automático de puertas según condiciones, plantas empaquetadoras, aparatos de maniobra de ascensores, etc.

-Robótica: control de los motores y captura de señales de los diferentes sensores.

Los microcontroladores de 16 y 32 bits integran en el propio chip **controladores de periféricos inteligentes** capaces de tratar procesos muy rápidos de una forma directa, la CPU principal supervisa a los microcontroladores «esclavos» que tratan las operaciones de bajo nivel, con lo cual, la potencia no solo esta en función de la frecuencia del reloj, sino que depende del número de unidades que se implementen en paralelo para trabajar en una estructura multiprocesadora.

Los microcontroladores de 16 bits, pueden hacer frente a las aplicaciones de los microcontroladores de 8 bits con más capacidad, pero fundamentalmente fueron diseñados para aplicaciones de **control de lazo cerrado**.

Estas aplicaciones consisten en supervisar y mantener la variable de salida del sistema que se desea controlar y en generar nuevos valores de entrada, calculados para mantener la variable de salida en su estado apropiado.

Ejemplos de estas aplicaciones son:

- Posicionamiento de la cabeza de lectura y escritura de un disco duro.

- Control de variables físicas en plantas industriales (temperatura, humedad, gases, etc.).

- Control de motores de máquinas de Control Numérico (CNC) y Robots.

Finalmente algunos fabricantes tienen a sus microcontroladores intérpretes de BASIC, incorporados en su memoria ROM interna, permitiendo dar respuesta a situaciones donde es preciso automatizar algún proceso de manera «fácil» por personal no especializado en el conocimiento y aplicación del código máquina o lenguaje ensamblador, pero con gran experiencia en el campo de la automatización y conocedores del lenguaje de programación de alto nivel como el BASIC. Además de este interprete, ofrecen herramientas de programación y «puesta a punto» autosoportadas, como también funciones de autómata programable en BASIC.

Debido al desarrollo dado a los microcontroladores, las herramientas de instrumentación han evolucionado permitiendo el diseño y mantenimiento de los equipos que llevan estos chips, obteniendo así, los más altos desempeños.

Frecuentemente, los microcontroladores se usan para reemplazar circuitos complejos que ordinariamente requieren muchos chips de bajo nivel o necesitan que una unidad central de proceso (CPU) maestra controle cada tiempo para activar el circuito. El

circuito de interface de un teclado claro ejemplo del uso de un microcontrolador de un chip. En un computador personal (PC), una media docena de chips (excluyendo unidades de entradas/salidas (I/O) codificando y decodificando) son necesarias para recibir y decodificar una serie de tiempos y porciones de datos que son dirigidos al teclado, este sistema de circuitos de bajo nivel puede ser reemplazado por un microcontrolador que sustituye por completo el circuito viejo, incorporando aspectos adicionales en ese computador.

2.2. Tipos de Microcontroladores

Generalmente todos los fabricantes de microprocesadores lo son de microcontroladores.

Algunos microcontroladores tiene limitadas sus capacidades, las aplicaciones a las que se destinen estos microcontroladores deberán adaptarse a éstas. De ahí que cada familia disponga de muchos modelos que se diferencian en espacio de memoria, número de I/O, tipo de temporizador, etc.

Las casas fabricantes de microcontroladores más conocidas en el mercado son: Motorola, Intel, Microchip, National Semiconductor, Zilog, Mitsubishi Electric, Rockwell, SGS-Thomson entre otras

Otros fabricantes además de diseñar sus propios microcontroladores, fabrican algunos de los miembros de la familia 51[1], por ejemplo, Siemens, Philips/Signetics, AMD, Fujitsu, NEC, OKI, etc.

La siguiente es una reseña de las características más importantes de algunas familias de microcontroladores existentes en el mercado:

2.2.1. Familia MCS-48

En la década de los 70, Intel comercializó esta familia. El elemento principal de esta familia es el 8048, que alberga en su interior una CPU de 8 bits, una memoria de sólo lectura (ROM) de 1K x 8 bits, una memoria de lectura y escritura (RAM) de 64 bytes, 27 líneas de I/O y un temporizador de 8 bits.

El éxito de esta familia es tan espectacular que todavía hoy se emplea en muchos productos habituales. Tal es el caso de los teclados destinados a computadores y terminales, los cuales incorporan el microcontrolador 8048 para realizar las tareas de exploración, decodificación y transmisión del código de la tecla pulsada hasta el procesador.

---

[1]La familia 51 hace referencia a los microcontroladores fabricados por **INTEL**

2.2.2. Familia PIC

2.2.2.1. Introducción

En 1976, General Instruments Microelectronics lanzó al mercado el microcontrolador PIC1650, basado en tecnología P-MOS, con una arquitectura Harvard, RISC[2] y que estaba diseñado con ROM interna.

En 1989, Arizona Microchip Technology (antes General Instruments) anunció la nueva familia con tecnología CMOS[3] y versiones con EPROM[4] para desarrollo y con memoria PROM programables una sola vez (OTP)[5] para producción.

Actualmente la familia PIC está conformada por dispositivos como:

2.2.2.2. PIC16C54, PIC16C55, PIC16C56 y PIC16C57

Los cuatro pueden tener encapsulados PDIP, PLCC, SOIC y CDIP[6] con ventana.

- Tienen un escalado de frecuencias máximas de 40 kHz, 4 MHz, 8 MHZ.

---

[2] La arquitectura RISC (Reduced Instruction Set Computer) se caracteriza por poseer un conjunto de instrucciones reducido y simple, cuyo objetivo es la interrelación entre el "software y el "hardware" que logre la máxima eficiencia del equipo.
[3] CMOS: Complementary metal-Oxide semicondutor
[4] EPROM: Memoria programable por impulsos eléctricos
[5] OTP (One Time Programable) o programables por una sóla vez
[6] Refierace a la página 59, tabla 3.4 para el significado de estas siglas.

- Las capacidades de EPROM/PROM son de 512xl2 (PIC16C54/C55), 1024xl2 (PIC16C56) y 2048xl2 (PIC16C57).
- La capacidad de RAM en los tres primeros es de 32 bytes y en el PIC16C57 llega a 80 bytes.

La estructura interna y su funcionamiento no cambian entre los microcontroladores con encapsulados de 18 patillas (PIC16C54/C56: 12 E/S) y de 28 patillas (PIC16C55 /C57: 20 E/S). Cada I/O puede programarse dinámicamente e independientemente como entrada o como salida a nivel de bit. Cada instrucción ocupa una sola palabra de 12 bits, por lo que la longitud de código es muy compacta. La tendencias futura será la aparición del microcontrolador de esta familia con mayores velocidades (20 MHz) y el cambio de estructura para incluir un convertidor análogo/digital (A/D), una UART[7], una memoria E^2PROM y el paso a los 16 bits. En el conjunto de 33 instrucciones del PIC hay instrucciones orientadas al byte o al bit de instrucciones de control y de manejo de periféricos. Estas instrucciones se pueden direccionar de tres formas: de forma directa, indirecta o relacional.

La figura 2.1 muestra el diagrama de bloques simplificado de los microcontroladores PlCl6C5X, y en ella se observa que el bus de datos de 8 bits está separado del de instrucciones de 12 bits Prácticamente en todos los microcontroladores, las

[7] UART: Universal Asynchronous Receiver Transmitter (Circuito electrónico que transmite y recibe datos en el puerto serie

instrucciones son leídas de la memoria y ejecutadas en forma secuencial. La família PIC, al tener estructura Harvard usa dos estados de búsqueda y ejecución de instrucciones, haciendo posible la ejecución de cada instrucción en un sólo ciclo de máquina, salvo las instrucciones GOTO, CALL y las de salto (modificación en el contador de programa), las cuales necesitan 2 ciclos.

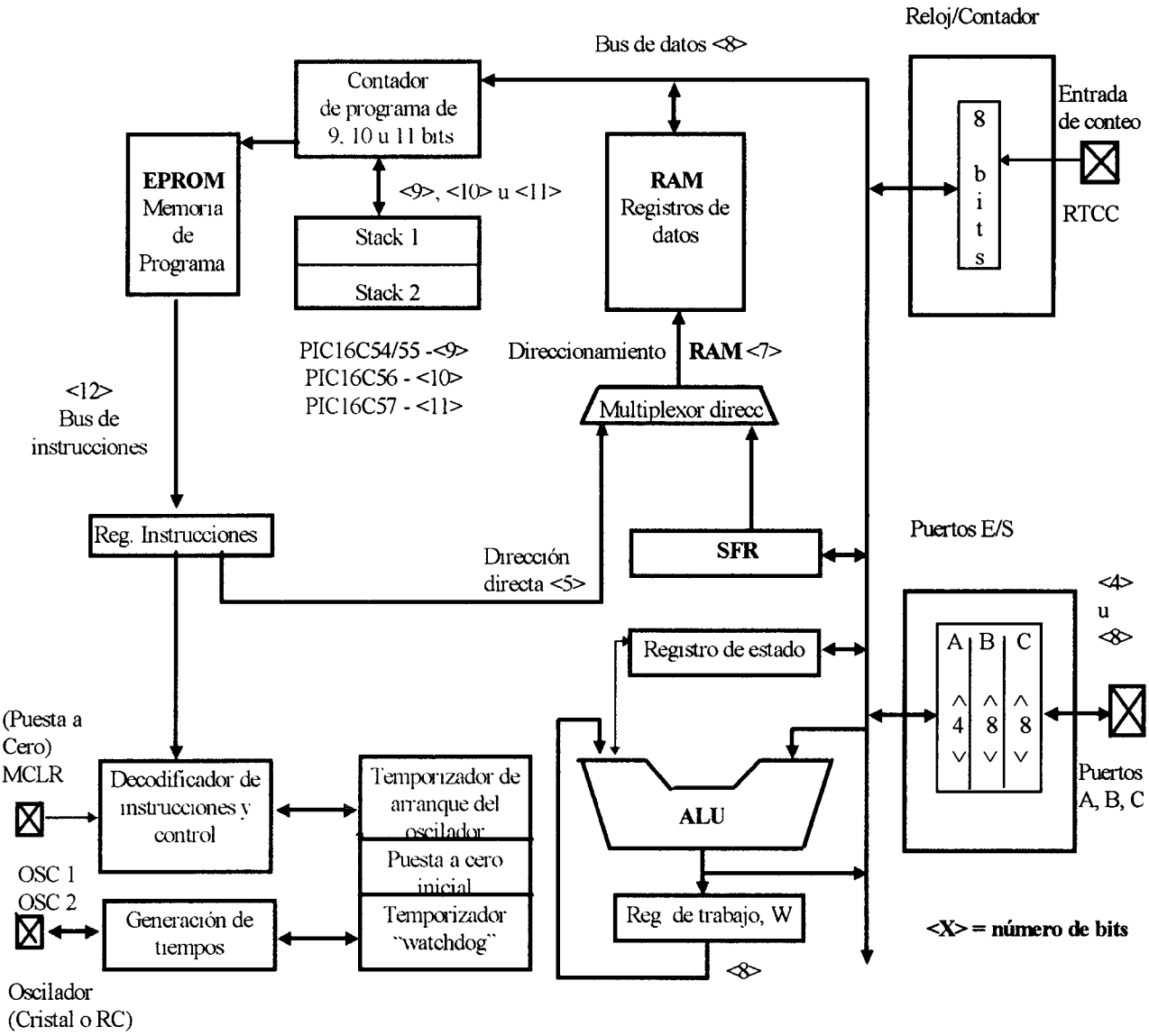


Figura 2.1. Diagrama simplificado de los microcontroladores PIC16C5X

Cada ciclo de máquina consiste en 4 ciclos reloj. La separación de los espacios de memoria de instrucciones y datos, permite cargar un valor de 8 bits de forma inmediata en un solo ciclo de máquina. Primero porque no hay conflicto entre la

búsqueda de la instrucción y la búsqueda del dato (en oposición a la arquítectura secuencial Von Neumann) y segundo, porque la palabra de la instrucción es suficientemente ancha para incorporar el dato de 8 bits. Dada su tecnología CMOS, el consumo es típico de 15 μA a 3 V y 32 kHz o es menor 2 mA a 4 MHz y 5 V. Cuando se le deja en reposo, después de ejecutar la instrucción SLEEP tiene un consumo típico de 2 μA. La memoria RAM mantiene la información con una tensión de sólo 1,5V. La tensión de alimentación puede variar entre 4 V y 5,5 V en la versión estándar y entre 3 V y 5,5 V en la versión de bajo consumo.

El oscilador puede realizarse con un circuito simple RC, o bien, con un cristal de cuarzo para una mejor estabilidad de frecuencia. La versión RC tiene un margen de CC a 4 MHZ, la low power (LP) de CC a 40 kHz, la XT de 400 kHz a 4 MHz y la futura high speed (HS) de 4 MHz a 8 MHz (2 MIPS).

La capacidad de programa interno varía desde 512 palabras hasta 2048 palabras de 12 bits en función de la versión elegida, y los registros internos disponibles pueden ser desde 32 hasta 80 registros de 8 bits. En la mayoría de los casos es suficiente para las aplicaciones a las que estos microcontroladores están enfocados.

En la velocidad se debe tener en cuenta que todas las instrucciones se ejecutan en un único ciclo de máquina (menos en las condicionales, en CALL y en GOTO que

necesitan 2 ciclos). Un ciclo de máquina consiste en 4 periodos de oscilador. En el caso de un oscilador de 20 MHz, el tiempo de un ciclo de instrucción es de 1/20/4 = 200 ns. En cuanto a la capacidad de memoria de programa, se debe tener en cuenta que todas las instrucciones de la familia PIC ocupan una sola palabra de 12 bits, con lo que para la versión de 512 palabras, se pueden disponer realmente 512 instrucciones.

En lo que respecta a las entradas/salidas, disponen de un puerto de 4 bits y uno o dos puertos de 8 bits según el encapsulado, todos ellos bidireccionales y programables dinámicamente bit a bit. Incorpora un circuito de vigilancia interno o "watch-dog" con un tiempo de 18 ms fácilmente retardable hasta 2,3 segundos, si se le asigna el contador interno ("prescaler").

La figura 2.1, presenta dos niveles de pila ("stack") con lo que el programa principal puede llamar a una rutina y ésta a su vez a una subrutina. Las aplicaciones típicas de estos microcontroladores son aquellas en las que se desea realizar funciones rápidas, sencillas y de bajo costo, tales como: sistemas de control para automóviles, controles de motores, sistemas de alarma, mandos a distancia, temporizadores digitales, codificadores y decodificadores, juguetes, telecomunicaciones, controles digitales de transceptores portátiles, instrumentación, etc.

El fabricante soporta a estos componentes con notas de aplicación de rutinas

matemáticas en coma fija y en coma flotante, controles de teclados y visualizadores, reemplazo de circuitos lógicos y circuitos lógicos programables (PLD), comunicaciones en serie, controles de humedad, detectores de humo, controles de luces, control de ratón, transmisor remoto, medida de resistencia, medida de temperatura, acceso a E^2PROM, entre otras.

2.2.2.3. PIC17C42-25

Es el microcontrolador de 8 bits más rápido del mercado. Diseñado con una avanzada arquitectura Harvard y múltiples prestaciones RISC, tiene una velocidad de ejecución de 6.25 MIPS (6.25 millones de instrucciones por segundo) varias veces mayor que la mayoría de los microcontroladores de 8 bits comparables. Dispone de una memoria E^2PROM de 2k * 16 y una RAM de 256 bytes. Puede trabajar en modo procesador direccionando directamente hasta 64 kpalabras de memoria de programa.

La versión OTP es ideal para una amplia gama de aplicaciones donde las prestaciones y la flexibilidad sean esenciales. Su elevada velocidad y versátiles periféricos hacen que sea muy adecuado para el control de motores de corriente continua como los que incorporan las impresoras, las máquinas de fax, las copiadoras, y para control industrial. Su bajo consumo, lo hace también válido para equipos alimentados con baterías, como por ejemplo instrumentación portátil.

2.2.3. Familia Motorola 6805

La familia 6805 es una de las más utilizadas en el mercado de los procesadores. Como ha sido optimizada para aplicaciones de control especializado, en lugar de procesamiento de datos, forma parte de dispositivos de producción masiva como juguetes, equipos de video, impresoras, modems y electrodomésticos. Hay 34 microcontroladores en la familia 6805. Cada año aparecen nuevos modelos que reemplazan a los anteriores. Todos estos dispositivos están construidos a partir de la misma CPU de 8 bits y tienen RAM, ROM, puertos de Entrada/Salida y temporizadores; algunos tienen, además, puertos seriales, convertidores análogo-digitales y memorias E^2PROM o EPROM.

La figura 2.2 muestra el diagrama de bloques del microcontrolador MC 68705P3; uno de los más simples de la familia. Una característica importante, es la ausencia de buses de datos y direcciones externos que habiliten la conexión de memorias RAM y ROM por fuera del microcontrolador. El contador de programa (PC) es de 12 bits, por lo tanto el máximo espacio de direccionamiento es de 4Kbytes (000 a FFF en el sistema hexadecimal de numeración).

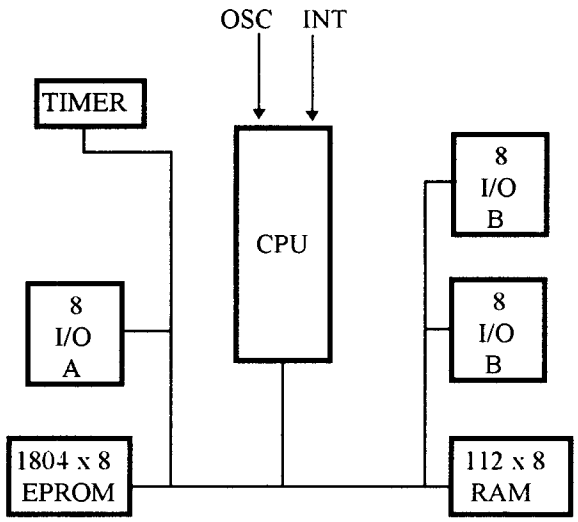


Figura 2.2. Estructura mínima del μC

Como el espacio de memoria y la velocidad de ejecución son críticos en aplicaciones de control, los programas deben escribirse directamente en lenguaje ensamblador. El modelo de registros, para la programación de la familia 6805, es uno de los más simples y eficientes. El acumulador A es un registro de 8 bits de próposito general que se utiliza para operaciones aritméticas y manipulación de datos. El registro X se emplea para direccionamiento con índice o como un acumulador auxiliar.

2.2.4. FAMILIA ZILOG Z8

2.2.4.1. CARACTERÍSTICAS

El Z8 aparece en 1979, introducido como una sofisticación en la arquitectura de los microcontroladores en un sólo chip, comparado con versiones anteriores.

El Z8 ofrece como características, ejecuciones rápidas (ciclos de instrucciones de 1.5 μs), uso eficiente de memoria, interrupciones vectorizadas , lineas de I/O, capacidad para manejo de bits y fácil expansión de la memoria.

En el Z8 podemos encontrar las características básicas de los otros microcontroladores como son:

- Hasta 20 Kbytes de memoria ROM interna o memoria RAM.
- 144 registros de 8 bits.
- Manejador de reloj, el cual se le puede conectar directamente un cristal controlador.
- 32 líneas de I/O
- Líneas de I/O serie

Además de las características generales, el Z8 incluye:

- 2 Contadores/Temporizadores de 14 bits
- 6 interrupciones vectorizadas
- UART para comunicaciones serie.
- Funciones de STACK interno ó externo.
- Opciones de apagado (Power Down)
- Compactibilidad TTL
- Grupo optimizado de instrucciones
- Interprete compilador/depurador para Basic y Forth

La famila Z8 está compuesta por una serie de microcontroladores con un chip, los

cuales son una variación del microcontrolador básico Z8601, Z8611, pasando por versiones de 18 pines como el Z8608, microcontrolador sin ROM, de bajo costo como el Z8681 y Z8682, microcontroladores emuladores como el Z8603 y el Z8613, versiones para desarrollo como el Z8612 y el Z8671 el cual trabaja en Basic, para terminar se tiene el Z8875 como compilador de Forth.

Entre las aplicaciones tenemos:

- Controladores de Disco.
- Controladores de impresora.
- Terminales
- Modems
- Controladores para Mouse
- Controladores Industriales
- Conmutadores telefónicos
- Instrumentación inteligente
-Mecanismos Automotrices.

cuales son una variación del microcontrolador básico Z8601, Z8611, pasando por versiones de 18 pines como el Z8608, microcontrolador sin ROM, de bajo costo como el Z8681 y Z8682, microcontroladores emuladores como el Z8603 y el Z8613, versiones para desarrollo como el Z8612 y el Z8671 el cual trabaja en Basic, para terminar se tiene el Z8875 como compilador de Forth.

Entre las aplicaciones tenemos:

- Controladores de Disco.
- Controladores de impresora.
- Terminales
- Modems
- Controladores para Mouse
- Controladores Industriales
- Conmutadores telefónicos
- Instrumentación inteligente
-Mecanismos Automotrices.

El mapa de direcciones del Z8 está conformado por los siguientes registros:

- Registro activo de la CPU para propósitos generales, especificos, control y registros generales.
- La memoria de programa de la CPU, la cual puede tener códigos de operación o datos.
- La memoria de datos, en la cual sólo se pueden almacenar datos.

Los registros pueden ser accesados como registros individuales de 8 bits o como pares de registros de 16 bits, usando direccionamiento directo o indexado. Todos los registros pueden ser usados como acumuladores, indices o apuntadores

La memoria de programa del Z8 posee un espacio de 64 K, de los cuales una porción es interna, dependiendo del microcontrolador. La memoria de programa esta dividida en una sección interna de 2k para Z8601, Z8603, Z8671, 4K para Z8611, Z8613 y una sección de 0 K para los microcontroladores sin ROM. (ROMLESS) Z8681, Z8682, Z86(91).

2.3. SELECCION DEL MICROCONTROLADOR

Cuando se opta por un microcontrolador, dentro de una amplia gama de posibilidades, es necesario tener en cuenta algunos aspectos, de los cuales se destacan los siguientes.

Hay que partir del hecho de que en el mercado existen fabricantes y distribuidores que cuentan con una amplia oferta de microcontroladores, los cuales cubren casi todas las expectativas de los clientes.

Antes de analizar los microcontroladores: en primer lugar, las características del sistema a controlar en aspectos tales como las *variables de entrada y/o salida del sistema análogas o digitales, además el tamaño del sistema*, y a un nivel mucho más específico variables como: *memoria, velocidad, comunicaciones con otros sistemas* entre otras.

También es indispensable, establecer algunos elementos contextuales, es decir, el medio dentro del cual se desarrolla el proyecto, porque a nivel tecnológico, no siempre lo más moderno es lo más apropiado. En este orden de ideas, es necesario establecer *las posibilidades de acceso a sistemas de desarrollo* y su disponibilidad en el mercado: *emulación, lenguajes de alto nivel, ensamblajes, facilidades de depuración*, como las más representativas; *documentación aportada por el cliente*; *servicio al cliente, en sus distintas modalidades.*

Lo que se quiere enfatizar en el comentario anterior es la importancia en la estructuración de todas las decisiones del proyecto en cuanto a la *coherencia entre la parte técnica, la financiera, la zona geográfica y la satisfacción de los*

requerimientos del cliente en sus necesidades actuales, dentro de *un diseño flexible que permita futuras ampliaciones o modificaciones.*

2.4. HERRAMIENTAS DE DESARROLLO

De manera similar a los sistemas utilizados con los microprocesadores para escribir, ensamblar y depurar programas en lenguaje de máquina, se requiere un sistema de desarrollo para cada familia de microcontroladores, el cual está compuesto por un paquete de software con editor, ensamblador y simulador de programas y, al mismo tiempo, se necesita de un hardware para "quemar" la memoria EPROM del microcontrolador

Un diseño con microcontroladores, en el que intervienen diferentes etapas (análisis del problema, generación del programa, verificación del producto, etc.), debe abordarse como una actividad estructurada que permita utilizar en cada momento las herramientas disponibles La figura 2.4 muestra la secuencia temporal de fases de un proyecto de este tipo.

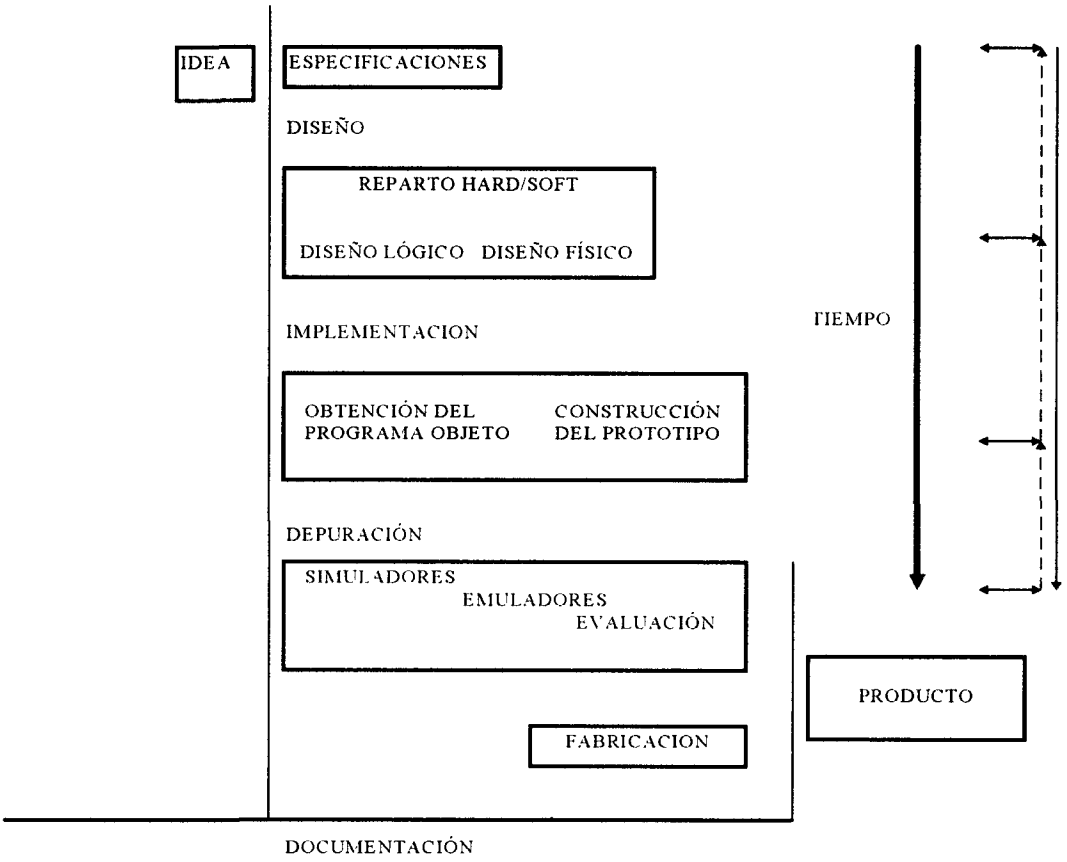


FIGURA 2.4. Fases de un Proyecto con Microcontroladores

Como se observa en la figura 2.4, la primera tarea consiste en el reparto de funciones a ejecutar (especificaciones) entre la componente software (programa) y la componente hardware (circuito) del sistema final. En la fase de diseño lógico se determina la estructura y seguridad de funcionamiento del programa, el origen y destino de los datos a manipular, y el tratamiento dado a las interrupciones. El diseño físico, por su parte, fija los componentes a utilizar y su interconexiones.

La fase de implementación consiste básicamente en la obtención del programa objeto.

Para ello, se necesitan una serie de herramientas software que, de forma resumida, serían:

- Editor de programas, que genera códigos fuente en ensamblador o en Lenguajes de alto nivel (C, Pascal, PLM, etc.).
- Ensamblador o compilador para lenguajes de alto nivel (ambos obtienen código objeto reubicable).
- Enlazador ("linker") de programas objeto y librerías específicas del microprocesador utilizado, que genera el código máquina final con direcciones absolutas.

La depuración consiste en la prueba del software sobre el hardware y, cada vez en mayor medida, del hardware como receptor y generador de interferencias electromagnéticas.

Pese a esta descripción secuencial de actividades, y aunque la figura 2.4 puede hacer pensar en etapas independientes de duración finita, el desarrollo de sistemas con microcontroladores siempre es un proceso interactivo, en el que las decisiones adoptadas en cada fase pueden afectar no sólo a las fases posteriores, sino también a las precedentes.

Estas exigencias de realimentación entre fases justifican, si no lo hiciera ya la propia

complejidad del problema a resolver y la necesidad de reducir sus costos, el empleo de herramientas integradas que permitan la puesta a punto rápida y efectiva del sistema.

Al conjunto de herramientas hardware + software que intervienen en la realización del proyecto se le denomina, en general, Sistema de Desarrollo. Estas herramientas son, en resumen:

- **Herramientas hardware**

- Ordenador y periféricos (impresoras, discos, etc).

- Emulador (microcontrolador + memoria + comunicaciones).

- Grabador de PROMs/EPROMs.

- Analizadores de Estados.

- Tarjetas de comunicaciones entre ordenador y emulador/grabador.

- **Herramientas software**

- Sistema operativo de emulación y editor de textos en ordenador.

- Ensambladores, Compiladores, Librerías y Enlazadores.

- Software de emulación, residente en el emulador, con traza[9], puntos de parada, control de memoria, comunicación con terminal, etc., y, eventualmente, depuradores interactivos con simbólicos, ampliaciones del sistema residente que permiten la

[9]Se entiende por traza el registro de estados internos y líneas E/S capturadas durante los pasos de programa ejecutados por el emulador.

emulación simbólica, más transparente para el usuario que la elemental con direcciones absolutas.

De las herramientas de desarrollo anteriormente descritas, se describe a continuación el emulador :

Un emulador es un equipo basado en varios microcontroladores que permite simular en tiempo real el comportamiento de hardware y software del sistema microcontrolador que deseamos construir. Emular significa, por tanto, simular el comportamiento de dicho sistema utilizando total o parcialmente el hardware de un emulador.

En general, el emulador hace el papel de interfaz entre el sistema microcontrolador que se desea probar y un ordenador, a través del cual se generan las órdenes y se recogen los resultados de la emulación.

Como ejemplo de emulador tenemos el MICE II (Microtek In Circuit Emulator) que es una herramienta de emulación universal que puede emplearse para cualquier familia de microprocesadores o microcontroladores de 8 o 16 bits, simplemente cambiando una tarjeta personalizadora para adaptarlo al microcontrolador deseado.

Las prestaciones del MICE II mejoran notablemente si se controla desde un

computador, con un software de emulación avanzado denominado USD (Universal Simbolic Debugger), que permite la depuración con símbolos, análisis de tiempos, presentación gráfica en forma de análisis lógico, edición de macrocomandos y otras funciones que facilitan enormemente el trabajo de emulación. Las características propias de este emulador son:

- Funcionamiento a la máxima frecuencia nominal del microcontrolador emulado.
- Mantenimiento del mapa de memoria y direcciones E/S de la placa prototipo durante la emulación.
- Habilitación/deshabilitación de señales de hardware desde el software de emulación.
- Ensamblado y desensamblado de código residente, incluyendo directivas de definición de espacio de memoria

Hay compañias como AVOCET SYSTEMS que cuentan con herramientas de desarrollado tanto para microprocesadores como para microcontroladores, ejemplo de ellas tenemos la de microprocesadores de la familia Z-80 y microcontroladores de la familia 8051 (avmac51).

También existe una herramienta que permiter desarrollar y depurar los programas para la familia PIC16C5X, la cual cuenta con un ensamblador (PICALC), un simulador (PICSIM), un grabador (PICPRO) y un emulador en tiempo real (PIC-ICE).

Lógicamente, por su tamaño no es posible trabajar con lenguajes de alto nivel, pero su conjunto de 33 instrucciones es suficientemente potente para desarrollar muchas aplicaciones con facilidad.

El PICALC es un ensamblador cruzado que se ejecuta en ordenadores con sistema operativo MS-DOS. La estructura del programa fuente consiste en nemónicos, directivas, macros, simbolos, etiquetas, constantes, expresiones y comentarios.

El PICSIM es un simulador cruzado que se ejecuta también sobre MS-DOS, permitiendo la simulación de la ejecución del programa con una capacidad de control de proceso que hace extremadamente fácil la depuración del programa fuente. El PICPRO es un grabador de la familia PIC que se controla mediante la conexión RS-232, permitiendo la transferencia de ficheros desde y hacia el PICPRO y realizando las funciones de test de desarrollo, grabación y verificación asi como "checksum", firma de identificación y control de bit especiales de selección del tipo de oscilador, la habilitación del "watchdog" y el de protección contra copia fraudulenta. El PIC-ICE es un emulador en tiempo real conectable al ordenador con MS-DOS, disponiendo de una sonda para su conexión al circuito que se desea emular.

3.0 EL MICROCONTROLADOR INTEL 8051

3.1 INTRODUCCIÓN

Indudablemente, el microcontrolador 8051 de Intel ha sido y continúa siendo, a través de sus derivados, uno de los más utilizados a la hora del diseño de aplicaciones con este tipo de dispositivos. Tanto es así, que de hecho se ha convertido en el estándar a nivel industrial adoptado en infinidad de aplicaciones. Tal vez haya contribuido a ello la acertada política de Intel en cuanto a la explotación de patentes y concesión de autorizaciones de segundas fuentes, las suficientes como para que el usuario no tenga problemas de suministro y se asegure una permanente disponibilidad de los chips en el mercado.

3.2. CARACTERÍSTICAS DE LA FAMILIA 51

Esta familia de microcontroladores contiene varios miembros, cada uno de ellos acondicionado para aplicaciones específicas. Todas las versiones existentes están

conformadas según el diagrama de bloques de la figura 3.1. Tienen la misma CPU, memoria RAM, temporizadores, puertos paralelos y entradas/salidas de tipo serial. Dentro de los modelos de la familia 51 de microcontroladores (ver tabla 3.1.) se encuentran los 8XX2 y 8XX1, que presentan las siguientes diferencias básicas :

- 8032/31 Memoria de programas externa.
- 8052/51 Memoria de programas interna en ROM.
- Memoria de programas interna en EPROM.

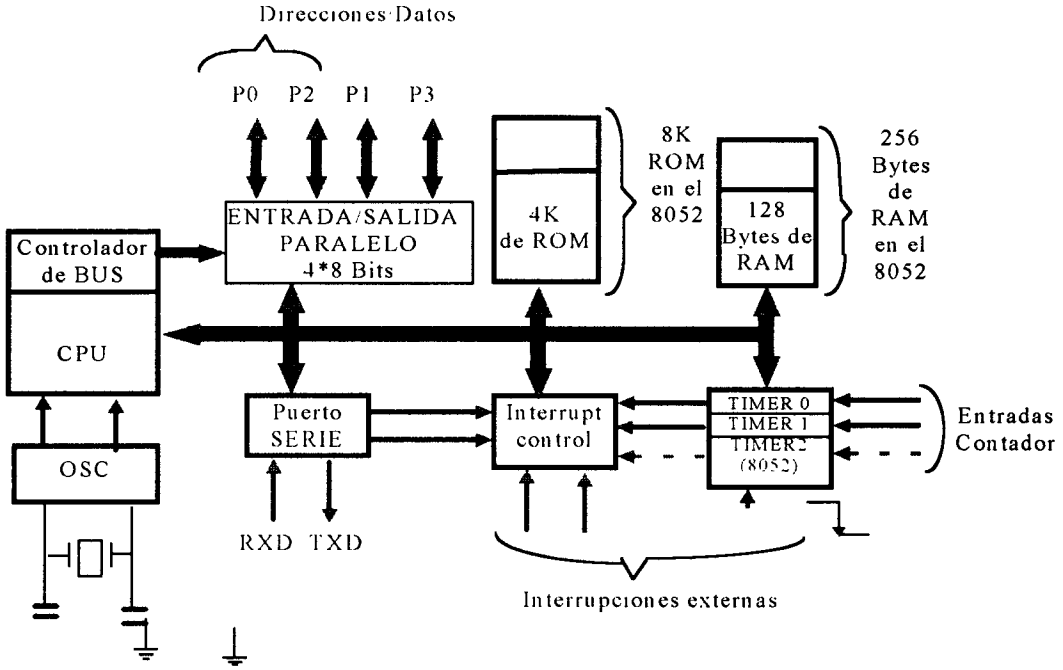


FIGURA 3.1. Diagrama de Bloques del microcontrolador 8051/8052

Tabla 3.1: Miembros de la familia de microcontroladores 8051

Nombre del dispositivo	Versión de ROM	Versión de EPROM	ROM Bytes	RAM Bytes	Temporizador 16-bit	Tecnología
8051	8031	(8751)	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

3.3. CARACTERÍSTICAS DE LOS MICROCONTROLADORES 8051

Es un microcontrolador de 8 bits de la familia **MCS-51** de **INTEL**. Sus caracteristicas principales son:

- 4 puertos de 8 bits con líneas bidireccionales, direccionables individualmente.
- 128 bytes de **RAM**.
- 2 Timer/Contadores de 16 bits.
- UART full duplex
- 5 vectores de interrupción.
- 4K bytes de **ROM**
- 64K para direccionamiento externo de memoria.
- 5 modos de direccionamiento para la ejecucion del set de instrucciones: directo, indirecto, por registro, inmediato e indexado (solo en **ROM**).

3.4. DESCRIPCIÓN DEL MICROCONTROLADOR 8051

El Microcontrolador 8051 posee toda una gama de registros y funciones, la cual determina su buen desempeño. Dichas funciones se describen en forma resumida.

La serie de registros que posee el microcontrolador son denominados Registros de Funciones Especiales (SFR), y se encuentran repartidos en el mapa de memoria del

microcontrolador ocupando una determinada posición de memoria. Dichos registros son:

B Registro B

ACC Acumulador A.

PSW Program Status Word (Palabra de Estado del Programa).

IP Prioridad de Interrupción.

SCON Control de Puerto Serial.

SBUF Buffer de Dato Serial.

P0 Control del Puerto cero (0).

P1 Control del Puerto uno (1).

P2 Control del Puerto dos (2).

P3 Control del Puerto tres (3).

TCON Control del Timer/Counter.

TMOD Modo de Control del Timer/Counter.

TL0 Control de Timer 0 LSB (Byte Bajo).

THO Control de Timer 0 MSB (Byte Alto).

TL1 Control de Timer 1 LSB (Byte Bajo).

TH1 Control de Timer 1 MSB (Byte Alto).

SP Stack Pointer 8 bits. (Puntero de la memoria).

DPL Apuntador de Datos LSB (Byte Bajo).

DPH Apuntador de Datos MSB (Byte Alto).

PCON Power Control (Control Consumo de Potencia).

El **registro B** es usado en operaciones de multiplicación y división.

El **acumulador A** es el registro Utilizado en operaciones lógicas y aritméticas.

El **PWS** contiene información sobre las banderas de Acarreo (Carry), sobreflujo, y paridad entre otras (contiene información de estado de la CPU en cada ciclo de instrucciones).

El apuntador de datos almacena los 16 bits de direcciones. Se puede manejar como un registro de 8 ó 16 bit en forma independiente.

El buffer de datos seriales en realidad son dos registro *buffer* fisicamente separados, uno para transmisión y otro para recepción.

Los registros del timer son pares de registros de 16 bits, que pueden actuar como temporizadores o contadores.

Los registros IP, IE, TMOD, TCON, SCON Y PCON contienen el control y estado de los bits de interrupción del sistema, de los Timer/Counter, y de los puertos seriales.

Los 4 puertos son bidireccionales. El P0 y el P2 se utilizan para accesar memoria externa (LSB y MSB respectivamente). Todos los pines de los puertos P1 y P3 son

multifuncionales.

El acceso de memoria externa es de dos modos:

- Programa de memoria externa (Almacenado en ROM).

- Datos de memoria externa.

Los Timer/Counter poseen diferentes formas de funcionamiento dependiendo de los registros de control. Sirven como generadores de rata de baudio, divisores de frecuencia para diferentes ciclos de conteos (preescaler), contadores de eventos, etc.

La interface serial se puede hacer de tres formas:

- Tx y Rx de 8 bits.

- Tx y Rx de 10 bits. 1 bit de Start y 1 bit de Stop.

- Tx y Rx de 11 bits. 1 Start, 1 Stop y 1 bit programable.

En este último la rata de baudios es variable.

Se manejan 5 fuentes de interrupción:

- 2 Externas (INT0, INT1).

- 2 del Timer.

- 1 de comunicación serial.

El registro IP permite categorizar las interrupciones dandole más importancia a unas que a otras.

Existen 2 versiones para los modos de operación del microcontrolador:

Versión HMOS ó de baja potencia y Versión CHMOS.

Al resetear el microcontrolador se escriben ceros (0) en todos los registros del SFR sin afectar la RAM interna.

3.5. DERIVADOS DEL MICROCONTROLADOR 8051

En el caso de la familia 8051, tanto Intel como sus segundas fuentes han contribuido a que proliferaran las más diversas soluciones integradas alrededor del corazón del 8051, considerado como una simple caja negra. A continuación presentamos algunas, no todas, de estas soluciones ofrecidas por diversos fabricantes, pero que tienen en común su punto de partida el 8051 de Intel.

3.5.1. El UCS51 de Intel

La familia UCS51 de Intel es una solución ASIC (Application Specific Integrated Circuit = circuito integrado específico) en la que el diseñador, con la ayuda del fabricante, puede reconfigurar las conexiones del núcleo del microcontrolador y unirlas a otros bloques funcionales que se adapten a los requerimientos específicos de la aplicación a desarrollar. Así, el cliente puede incorporar periféricos, puertos de E/S, conversores, etc., gracias a que en este caso concreto Intel ha hecho accesible el bus

Interno SFR (Special Function Register) del núcleo del μC. Esta nueva familia ofrece, además de las prestaciones básicas, cinco interrupciones externas que pueden añadirse a las existentes en el microcontrolador estándar. El cliente, puede incorporar a "su chip" cualquiera o varios de los siguientes bloques funcionales:

- El tamaño de la ROM: 0K, 4K, 8K ó l6K.
- El tamaño de la RAM: 128 ó 256 bytes.
- Número de interrupciones: 5 ó 10.
- Conversor analógico/digital (UCS51AD): ocho canales analógicos; precisión de 8 bits; tiempo de conversión de 20 μs a 16 MHz; linealidad total ±l LSB.
- Canal serie de comunicaciones (UCS5lSlO) con 5 modos posibles de operación.
- Temporizador/contador (UCS51T2) de 16 bits.
- Unidad de interfaz SFR (UCS5lBIU) que permite conectar al bus interno SFR periféricos desarrollados a partir de la biblioteca ASIC estándar (1,5 μCHMOS III).
- Generador de frecuencias en baudios (UCS5lBRG).

El proceso de implementación de este microcontrolador a medida se realiza con la ayuda de potentes herramientas informáticas que permiten realizar el dibujo del esquema inicial, la simulación del circuito, la verificación de las conexiones entre el núcleo UCS51 y la lógica periférico añadida por el cliente. Todo ello dentro del entorno IDE (Intel Design Environment), absolutamente transparente al usuario, que

puede ejecutar instrucciones del 8051 e imponer simultáneamente valores 1 o 0 en los terminales de entrada del circuito con objeto de comprobar, en las salidas o en el bus de datos, la validez de las conexiones.

La proliferación de circuitos con el corazón del 8051 es una posibilidad que ofrece el fabricante al cliente para implementar microcontroladores ajustados a sus necesidades. Así se beneficia de todo el software existente desarrollado específicamente para la familia 8051.

3.5.2. El C8-P31F-64

El C8-P3lF-64, de White Technology Inc., es un sistema microcontrolador basado en el estándar industrial 8051. Consiste en el núcleo del 8OC 31 con la adición, en el propio chip (de 40 patillas), de 64K bytes de memoria flash para programa (de OOOOH a FFFFH) y 8K bytes de RAM para datos (de 8OOOh a 9FFFh).

Este microcontrolador trabaja a 16 MHz con un tiempo de ciclo de escritura de 150 ns y un consumo muy reducido, al estar realizado totalmente con tecnología CMOS. La corriente máxima de alimentación a +5 V es de sólo 25 mA, que se ven reducidos a 200 μA en el modo de bajo consumo

El borrado y la programación de la memoria flash interna se realiza a través del Registro de Comandos (Command Register), cuando se aplica un nivel alto (12 V) a la patilla Vpp. El contenido del registro sirve como estado de entrada al estado de máquina interno, mientras que las salidas de este último determinan la función a desarrollar sobre la memoria.

En la tabla 3.2. se muestran los comandos que hacen referencia a la memoria flash.

Tabla 3.2. Comandos de la memoria flash.

COMANDO	CICLOS	CICLO 1		CICLO 2	
		OPER.	COD.	OPER.	COD.
Lect.mem.	1	WR	OOh	--	--
Borr./borr.	2	WR	2Oh	WR	2Oh
Borr./verif.	2	WR	AOh	RD	x
Prog./prog.	2	WR	4Oh	WR	x
Prog./verif.	2	WR	COh	RD	x
Reset.	2	WR	FFh	WR	FFh

El Registro de Comandos (en adelante CR) no ocupa una dirección de la memoria, sino que está realizado con básculas y se utiliza para almacenar el comando. Se puede escribir en el CR poniendo la señal WE a 0 mientras CE esté previamente a 0. Los tres bits más altos (R7, R6, R5) contienen el código de la función a ejecutar. Los otros bits están a 0 con excepción de la función de Reset que tiene por código FFh. Los bits R7-R0 del CR se corresponden con los D7-D0 del bus de datos.

Cuando se aplica un 0 a la patilla Vpp del chip, el contenido del CR pasa a ser OOh, permitiendo únicamente operaciones, de lectura de la memoria flash. Al aplicar

tensión al microcontrolador en la puesta en marcha, adopta por defecto el contenido OOh.

Cada uno de los comandos del CR desencadena el inicio de un algoritmo asociado, que ejecuta la acción solicitada. Veamos una somera explicación de dos algoritmos:

- **Algoritmo de programación.** -Tiene una duración aproximada de 10 µs por byte. A cada operación de escritura de un byte le sigue otra de verificación para comprobar que ha sido correcta. El bucle de escritura de un byte está formado aproximadamente por 25 instrucciones básicas del 805 l.
- **Algoritmo de borrado.**-*Su* duración es de aproximadamente dos segundos para el borrado de toda la memoria de 64K bytes. Se efectúa en dos ciclos mediante el código 2Oh para asegurar que el borrado no se producirá de forma accidental.

3.5.3. El DS5000 de Dallas Semiconductor

Esta versión del 8051 cuenta con notables ventajas sobre la versión original de Intel, siendo las más notables:

l. Incorpora en el chip una memoria RAM no volátil, tanto para programa como para datos, de 8K ó 32K bytes y con una autonomía en ausencia de alimentación de aproximadamente 10 años. Esta capacidad de retención de datos permite preservar información por largos periodos de tiempo y lo que es más, reflejar en tablas el

conocimiento acumulado del sistema de control desde su puesta en marcha inicial. Consecuentemente, nada impide añadir al sistema la habilidad de "aprender" de su experiencia y dotarlo de la capacidad de reaccionar alterando el desarrollo del programa en respuesta a los cambios de condiciones en largos periodos de tiempos. Resumiendo, es capaz de hacer lo mismo que hasta hace bien poco estaba reservado a grandes ordenadores. Como es obvio, la tecnología utilizada es CMOS, lo que redunda en general sobre el consumo del chip: 25 mA en funcionamiento normal y unos 200 μA en régimen de bajo consumo.

2. Detección del funcionamiento errático del programa, con desencadenamiento automático de una nueva puesta en marcha (Reset).

3. Carga inicial del programa en el microcontrolador por medio del canal serie o paralelo de un sistema de mayor envergadura (generalmente un PC compatible).

4. También dispone del software interno necesario para trabajar con programas y datos cifrados, totalmente ininteligibles al observador externo. La llave utilizada es de 40 bits, lo que lo hace inexpugnable a los piratas, protegiendo la inversión que representa el programa interno y/o los datos.

5. Absoluto control de datos y programa en las operaciones de puesta en marcha y

paro.

Como era de esperar, el microcontrolador de Dallas es absolutamente compatible pin a pin con el 8051 de Intel y otro tanto a nivel de compatibilidad de instrucciones. La propia firma dispone de otras implementaciones del 8051 con reloj/calendario incorporado, con memorias de 8, 32, 64 o 128K bytes configurables de forma variable, con conversores A/D de 12 bits, etc.

3.5.4. La saga 8051 de Philips/Signetics

Tal vez sea Philips/Signetics el fabricante que más derivados del 8051 dispone (Ver las tablas 3.3. y 3.4. proporcionadas por el fabricante).

Todos los derivados presentados tienen la característica común de estar fabricados con tecnología CMOS, lo que les confiere las propiedades inherentes a dicha tecnología: escaso consumo y una buena inmunidad al ruido (reforzada en algún caso).

Nótese también en las tablas 3.3 y 3.4 que el código de identificación de todos los microcontroladores empieza por 8, cosa que no se nos hace extraña si consideramos que casi todos los chips del primer fabricante del 8051 (Intel), suelen empezar por esta cifra (8008, 8085, 8088, 80286, etc.). La segunda cifra, representada en las tablas 3.3 y 3.4 generalmente por una X, puede variar e informa a su vez de algunas

características especiales o diferenciadoras del μC considerado. En la tabla 3.5 se muestra el significado que Philips/Signetics atribuye a este número. Lamentablemente no se trata de una nomenclatura estándar y no todos los fabricantes de derivados del 8051 la han adoptado, aunque la mayoría de las veces suele parecerse.

Al describir los microcontroladores ASIC de Intel, veíamos la gran diversidad de derivados que se podían obtener: uno para cada caso concreto, para cada caso a resolver. Evidentemente, no siempre es posible llegar a esta solución, dado que el costo inicial de diseño es muy elevado y no es rentable si no es fabricando en serie. Para resolver situaciones en las que se precisen pocas unidades en total, lo más razonable es utilizar un derivado del 8051 (si nos hemos decidido por este microcontrolador), que además del corazón del mismo integre ya alguno o todos los bloques funcionales periféricos que nos interesen.

Tabla 3.3. Características de los derivados del 8051 de Philips-Signetics

MICRO	OTP &	ROM	RAM	PUERTOS	E/S SERIE		TEMP.	A/D		INTERRUPCIONES	
CONTROLADOR	EPROM	(Bytes)	(Bytes)	8-BIT	UART	I^2C	16 bit	BITS	CANALES	INT	EXT
8XC751 (1)		2K	64	2+3/8		•	1			3	2
8CX752 (1)	•	2K	64	2+5/8			1	8	5	4	2
80C31/8XC51 (2)	•	4K	128	4	•		2			3	2
80CL31/80CL51	(3)	4K	128	4	•		2			3	2
8XCL410	(3)	4K	128	4		•	2			3	10
8XC851		4K	128	4	•		2			3	2
8XC550	•	4K	128	4	•		2+WD	8		4	2
8XC451	•	4K	128	7	•		2			3	2
8XC852 (1)		6K	256	2/8			2			5	1
8XC652	•	8K	256	4	•	•	2			4	2
8C32/8XC52 (2)	•	8K	256	4	•		3			4	2
8XC562	(4)	8K	256	6	•		3+WD	8		9	6
8XC552	•	8K	256	6	•	•	3+WD	10		9	6
80C053 (1)	(5)	8K	192	3+4/8			2			3	2
8XC054 (1)	•	16K	192	3+4/8			2			3	2
8XC654	•	16K	256	4	•	•	2			4	2
8XC592	•	16K	512	6	•		3+WD	10		9	6
8XC524 (1)	•	16K	512	4	•	•	3+WD			5	2
8XC528	•	32K	512	4	•	•	3+WD			5	2

(1) No dispone de derivado con ROM externa
(2) 80C31, 80C32 = ROM externa.
80C51, 80C52 = ROM por máscara
87C51, 87C52 = EPROM/OTP.
(3) Disponible con ROM externa sobre el chip (Piggyback)
(4) Usar 87C552 EPROM/OTP
(5) Usar 87C054 EPROM/OTP.

Tabla 3.4. Características de los derivados del 8051 de Philips-Signetics

MICROCONTROLADOR	ENCAPSULADOS	CARACTERISTICAS DESTACABLES
8XC751 (1)	A28, F24, N24	Bajo costo. Encapsulado de 24 patillas.
8XC752 (1)	A28, F28, N28	PWM.
80C31/8XC51 (2)	A44, B44, F40, K44, N40	Estándar industrial.
80CL31/80CL51	N40, D40	Baja tensión/consumo (1,8 a 6V).
80CL410	N40, D40	Baja tensión/consumo (1,8 a 6V).
8XC851	A44, B44, N40	256 bytes E^2PROM.
8XC550	A44, B44, F40, K44, N40	WD.
8XC451	A68, F64, K68, N64	7 puertos de E/S.
8XC852 (1)	Tarjeta	2K de E^2PROM (datos y programa).
8XC652	A44, B44, F40, K44, N40	8K de ROM.
80C32/8XC52 (2)	A44, B44, F40, K44, N40	Estándar industrial
8XC562	A68, B80	2PWM, WD, T2
8XC552	A68, B80, K68	2PWM, WD, T2
83C053 (1)	N42	9PWM, 3 entradas A/D por software, OSD
8XC054 (1)	N42	9PWM, 3 entradas A/D por software, OSD
8XC654	A44, B44, F40, K44, N40	16K ROM. Mejorado frente EMC.
8XC592	A68, K68, B80	CAN bus, WD.
8XC524 (1)	A44, B44, F40, K44, N40	16K ROM, 512 RAM, WD
8XC528	A44, B44, F40, K44, N40	32K ROM, 512 RAM, WD

TIPOS DE ENCAPSULADO

A=PLCC=Plastic Leaded Chip Carrier. B=PQFP=Plastic Quad Flat Pack.
D=VSO=Plastic Very Small Outline Package. F=CDIP=Windowed Ceramic Dual In-line Packege.
K=CLCC=Windowed Ceramic Leaded Chip Carrier. N=PDIP=Plastic Dual In-line Package.

Tabla 3.4. Significado de la segunda cifra.

8XCXXX

0	- ROM/EPROM externa.
3	- ROM interna grabada en fábrica.
5	- EPROM externa sobre el chip (piggyback). (*)
7	- EPROM/OTP interna.
8	- Versión E2PROM (PROM borrable eléctricamente). (*)

(*) Convención aún no adoptada por Philips/Signetics.

Para tener una idea más aproximada de la potencia y posibilidades que ofrece cualquiera de los derivados actuales del 8051, seguidamente se describirán las

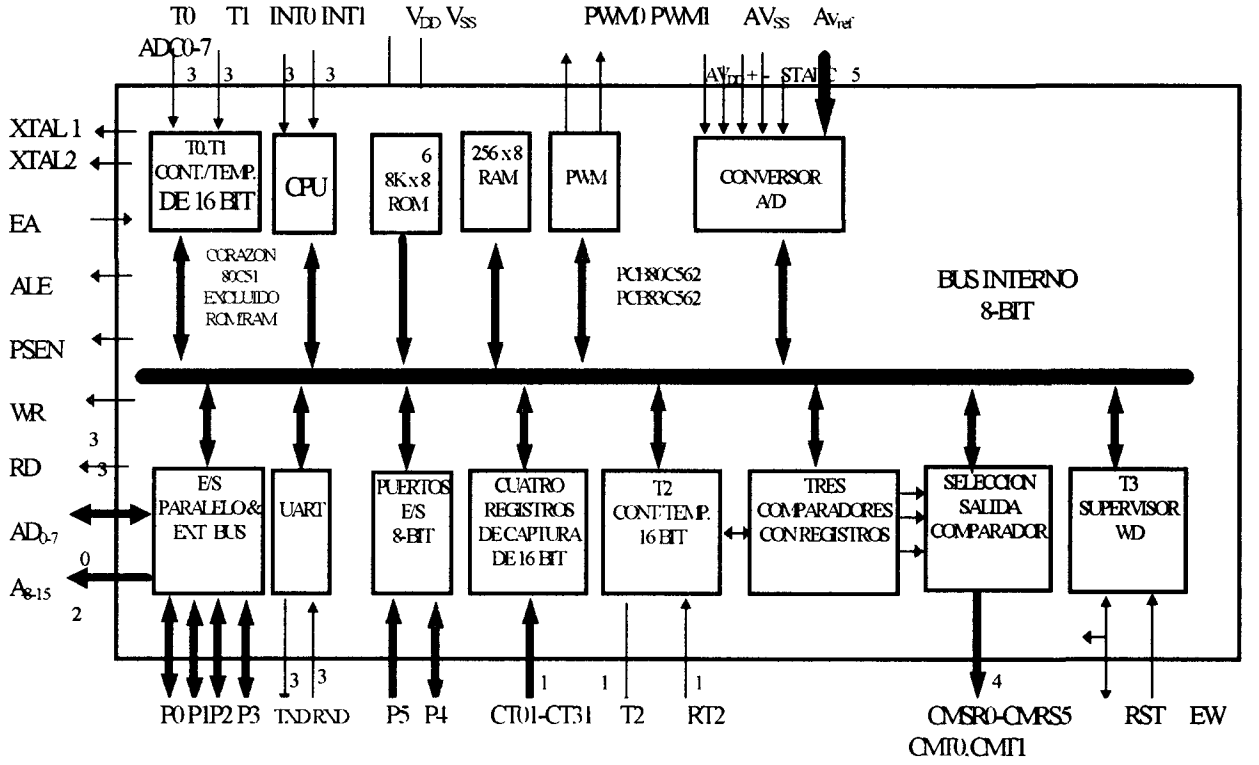
características más importantes de los tipos 8OC562 y 8OC552, que difieren en muy pocos aspectos.

A grandes rasgos, el 8OC562 (nos referiremos siempre a él aunque implícitamente también nos refiramos al 8OC552) cuenta, además del núcleo o corazón del 8051, con las siguientes funciones complementarias integradas sobre el mismo chip:

- 8 canales de conversión A/D de 8 bits (10 bits para el 8OC552).
- 2 canales PWM (resolución de 8 bits).
- 1 circuito de supervisión (watchdog).
- 1 temporizador/contador adicional T2.
- 1 canal serie I^2C (sólo en el caso del 8OC552).
- 1 UART compatible con la estándar 805 l
- 5 puertos de E/S reconfigurables.

En conjunto, es potente, compacto, de pequeñas dimensiones y muy equilibrado en la relación precio/prestaciones. Sus campos de aplicación son muy diversos: la robótica, la domótica, el automóvil, los controladores industriales, los periféricos de ordenador, etc.

En la figura 3.2 se muestra un diagrama de bloques del microcontrolador



0 FUNCION ALTERNATIVA AL PUERTO 0 3 FUNCION ALTERNATIVA AL PUERTO 3
1 FUNCION ALTERNATIVA AL PUERTO 1 4 FUNCION ALTERNATIVA AL PUERTO 4 6 NO EN EL PCB80C562
2 FUNCION ALTERNATIVA AL PUERTO 2 5 FUNCION ALTERNATIVA AL PUERTO 5

FIGURA 3.2. Diagrama de Bloques del 80C562

3.5.4.1. El bloque de conversión A/D de 8 bits

El sistema de conversión adoptado por el fabricante es el ampliamente empleado de aproximaciones sucesivas. Proporciona una relación aceptable entre complejidad y velocidad de conversión: 24 ciclos máquina, 24 µs a 12 MHz en el caso del 8OC562 y de 50 ciclos, 50 µs en el 8OC552 (pero recordemos que sobre 10 bits). En caso de no

utilizarse el bloque de conversión A/D, sus entradas pueden ser aprovechadas como un puerto paralelo de 8 bits.

El desencadenamiento y control del proceso de conversión se realiza mediante la intervención de los registros especiales SFR. Un bit de interrupción indica al sistema que la operación de conversión ha finalizado.

3.5.4.2. Las salidas de Impulsos Modulados en Anchura (PWM)

El microcontrolador dispone de dos generadores de impulsos PWM. La precisión de la modulación es de aproximadamente 1/250 ya que el valor se determina mediante un contador de 8 bits (256 valores). En el 8051 y sus derivados la participación de los registros especiales SFR será imprescindible para, en este caso, determinar la frecuencia de salida que podrá variar entre 92 Hz y 23,5 kHz aproximadamente (reloj a 12 MHz).

Estas salidas de impulsos pueden controlar directamente motores paso a paso o incluso utilizarse para la conversión D/A mediante la ayuda de filtros pasa bajo que conviertan las variaciones de los impulsos en una variación de tensión analógica.

3.5.4.3. Sistema de supervisión o watchdog

Está formado por un contador de 19 bits dividido en dos partes, una de 11 bits no accesible por el usuario, y otra de 8 bits programable externamente que permite definir un intervalo de supervisión de unos 2 ms a 0,5 s (a 12 MHz).

Su funcionamiento es sencillo Cíclicamente, antes de que se produzca un desbordamiento de capacidad del contador, por medio del programa, el usuario debe recargar el contador. Si se produce el temido desbordamiento, es señal de que el flujo del programa no ha llegado, por alguna razón (bucle infinito, retomo equivocado, etc.) a la secuencia de recarga, estratégicamente situada por el diseñador. En este caso, lo único que se puede hacer es volver a empezar, que es lo que hace automáticamente el bloque funcional de supervisión WD, que genera un RESET al llegar al final de la cuenta.

3.5.4.4. El temporizador/contador T2

Además de los contadores/temporizadores T0 y T1, incluidos en el 8051, los modelos 8OC552 y 8OC562, como otros de la misma familia, cuentan con un tercer temporizador/contador: el T2, algo más sofisticado y versátil que los mencionados anteriormente.

Está constituido por un contador de 15 bits asociado a cuatro registros de captura (CT0 - CT3) de 15 bits y a tres registros de comparación (CM0 CM2) de 16 bits. Como en casi todos los contadores/temporizadores, estos recursos pueden ser combinados para que el contador se active o no, además de poder utilizar como impulsos de contaje la fuente interna de reloj o una fuente externa por medio de la patilla T2.

3.5.4.5. El Canal de Comunicaciones I^2C del 8OC552

Puede que ésta sea la diferencia más notable con el 8OC562: la incorporación de un bloque funcional capaz de dar soporte a un canal de comunicación I^2C. El canal I^2C (Inter IC Communication) es un bus de intercomunicación serie propio de Philips/Signetics, muy difundido en el mercado y utilizado para la interconexión de circuitos integrados o tarjetas. Este tipo de bus no sólo trata datos, sino que también gestiona comandos, lo que le confiere una gran potencia de tratamiento. Desde el punto de vista físico, la interconexión es extremadamente sencilla puesto que sólo se emplean para ello tres simples hilos: masa, reloj (SCL) y datos (SDA), los dos últimos bidireccionales y permiten transferir información entre dispositivos conectados al bus. La figura 3.3 muestra la disposición física del bus que permite la interconexión de varios circuitos integrados o tarjetas entre sí, ya que cada uno de ellos tiene una dirección individualizada y sólo responde cuando es invocado directamente por otro.

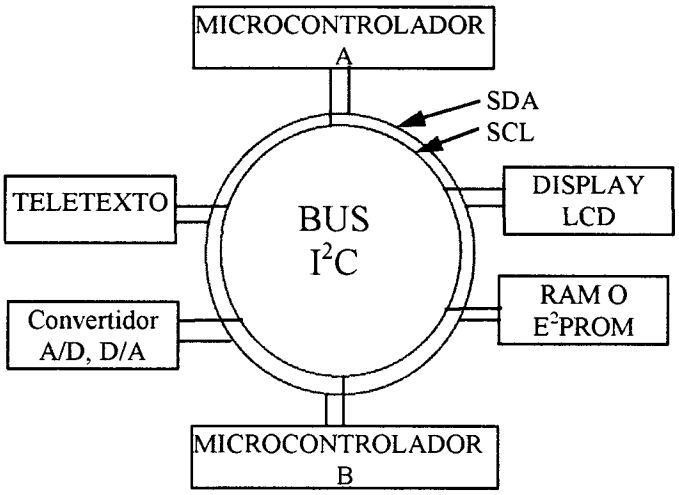


FIGURA 3.3. Configuración típica del bus I^2C

El bus I^2C debe disponer por lo menos de un dispositivo maestro, aunque puede tener varios, y de uno o varios esclavos que realicen las acciones de interfaz. Existe un protocolo de arbitraje del bus que determina quién y en qué momento podrá utilizarlo.

Como ventaja adicional existe una larga lista de circuitos integrados capaces de intercomunicarse por medio del bus I^2C: excitadores de visualizadores LED y LCD, puertos paralelo de E/S, relojes/calendario, conversores de datos A/D y D/A, memorias, circuitos especializados como decodificadores de teletexto, procesadores de vídeo o de deflexión, sintonizadores, telecomunicaciones, radio celular, etc.

3.5.4.6. Juego de Instrucciones

El juego básico de instrucciones es el propio del 8051, al cual se le han añadido funciones especiales de los SFR para el control de todos los bloques funcionales añadidos alrededor del núcleo del 8051. El conjunto de instrucciones consiste en 49 de un solo byte, 45 de dos bytes y 17 de tres bytes.

3.5.4.7. Organización de la Memoria

Su distribución no difiere mucho de la originaria del 8051. La CPU interna maneja operandos en tres espacios de memoria distintos.

La variante 83C552 dispone de 8K bytes de memoria ROM interna para programa, que pueden ser ampliados externamente a 64K bytes (en el 8OC 552 toda la memoria es externa). En cuanto a la memoria RAM, está dividida en tres secciones de 128 bytes: direcciones bajas de OOh a 7Fh y altas de 8Oh a FFh. La parte alta está solapada a su vez con los SFR (Special Function Registers). La forma de acceso a cada una de las secciones se hace con modos de direccionamiento distintos. La memoria RAM también puede ser ampliada externamente a 64K bytes.

3.6 ORGANIZACION DE LA MEMORIA

3.6.1 Memoria de Programa

Los 8051 tienen espacios separados de direcciones para la Memoria de Programa y la Memoria de Datos. La Memoria de Programas puede ser de hasta 64K bytes. La parte inferior, hasta 4K (8K para el 8052) y se encuentra interna en el Chip.

La figura 3.4 muestra un mapa de la memoria de programa del 8051.

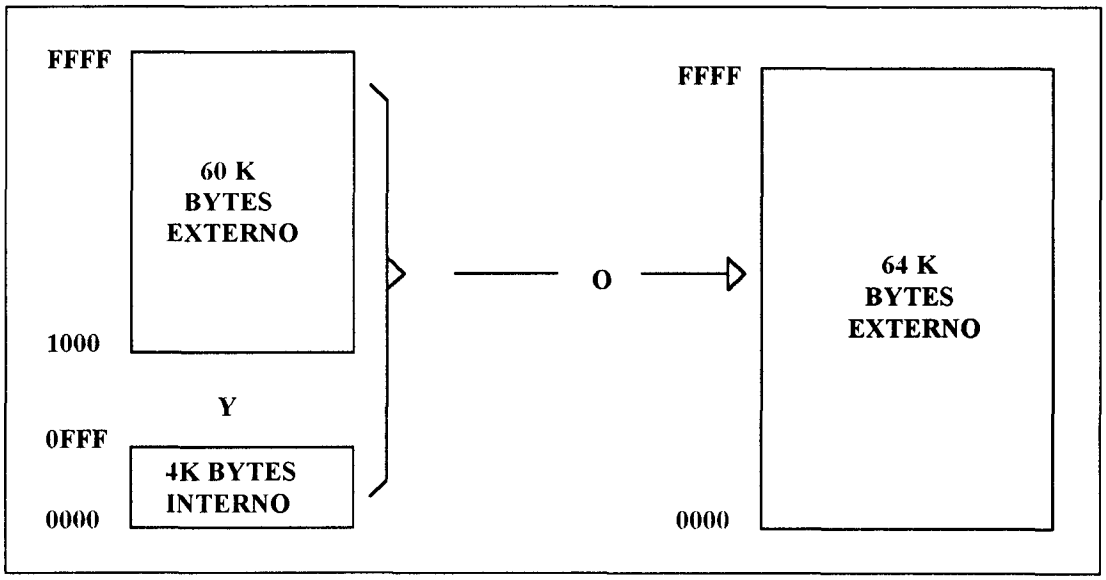


Figura 3.4. La Memoria de Programa del 8051.

3.6.2. La Memoria de Datos

Los 8051 pueden direccionar hasta 64 K bytes de Memoria de Datos. La instrucción "MOVX" se usa para acceder la memoria externa de datos. (Refiera al Set de Instrucciones del MCS-51, en este capítulo, para la descripción detallada de las instrucciones).

Los 8051 tienen 128 bytes de RAM en chip (256 bytes en los 8052) más un número de Registros de Funciones Especiales (SFRs). Los 128 bytes bajos de la RAM pueden ser accesados por direccionamiento directo (MOV data addr) o por direccionamiento indirecto (MOV @Ri). La figura 3.5 muestra la organización de la Memoria de Datos del 8051 y el 8052.

3.6.3. Área de Direccionamiento Indirecto

Note que en la Figura 3.5b. los SFRs y la dirección indirecta de la RAM tienen las mismas direcciones (80H-0FFH). No obstante, son dos áreas separadas y se acceden en dos maneras diferentes.

Por ejemplo: La instrucción **MOV 80H,#0AAH,** escribe 0AAH en el Puerto 0 que es uno de los registros SFRs y las instrucciones ,

MOV R0,#80H

MOV @R0,#0BBH

escribe 0BBH en la ubicación 80H de los datos de la RAM. Así, después de ejecutar las dos instrucciones anteriores, el Puerto 0 contendrá 0AAH y la posición 80H de la RAM contendrá 0BBH.

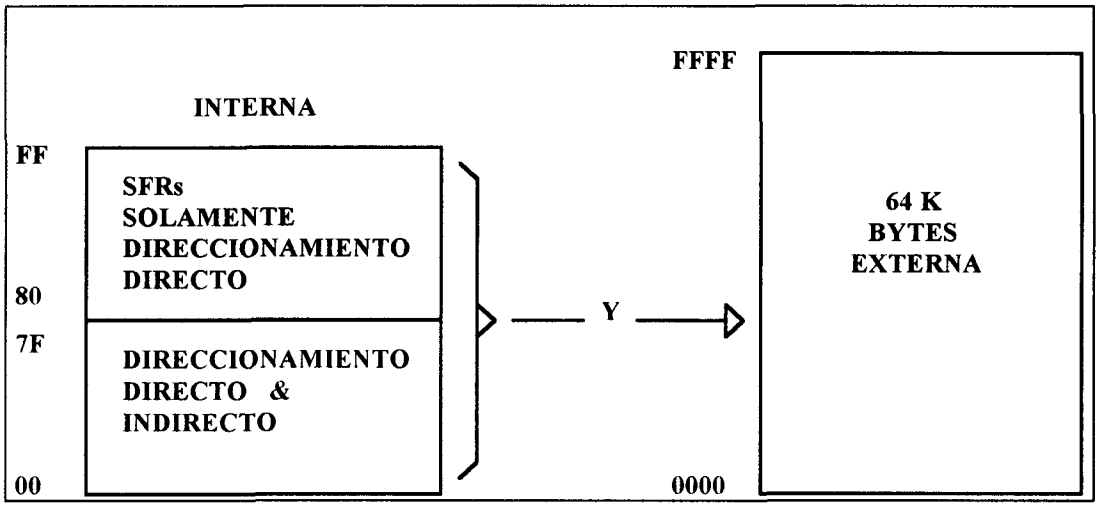


Figura 3.5a. Memoria de Datos del 8051.

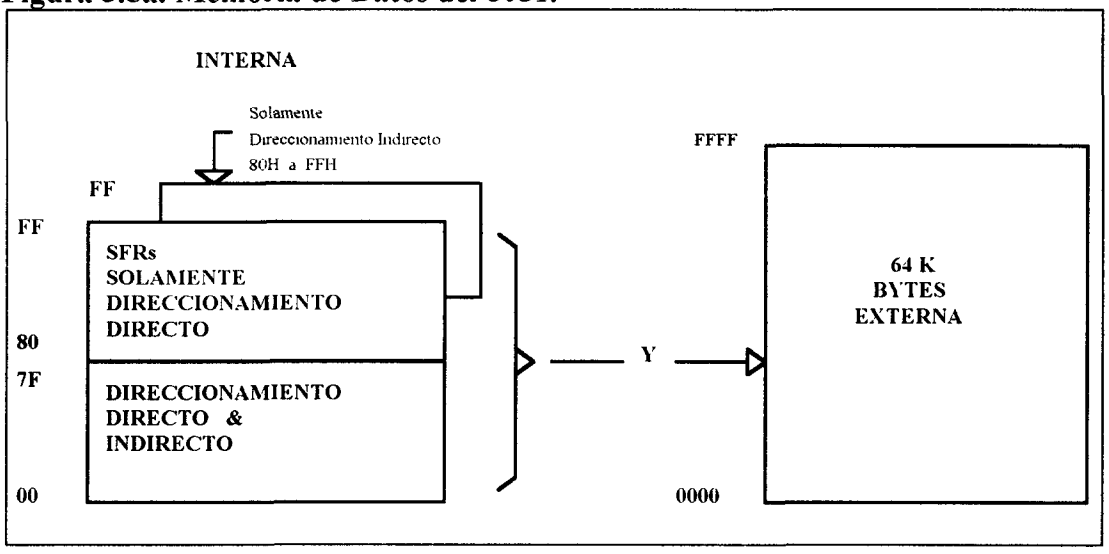


Figura 3.5b. Memoria de Datos del 8052.

3.6.4. Área de Direccionamiento Directo e Indirecto

Los 128 bytes de RAM que pueden ser accedidos por ambos direccionamientos (directo e indirecto) se puede dividir en tres segmentos como se enumera a continuación y se muestra en la figura 3.6

La figura 3.6 muestra los diferentes segmentos dentro del chip de la RAM.

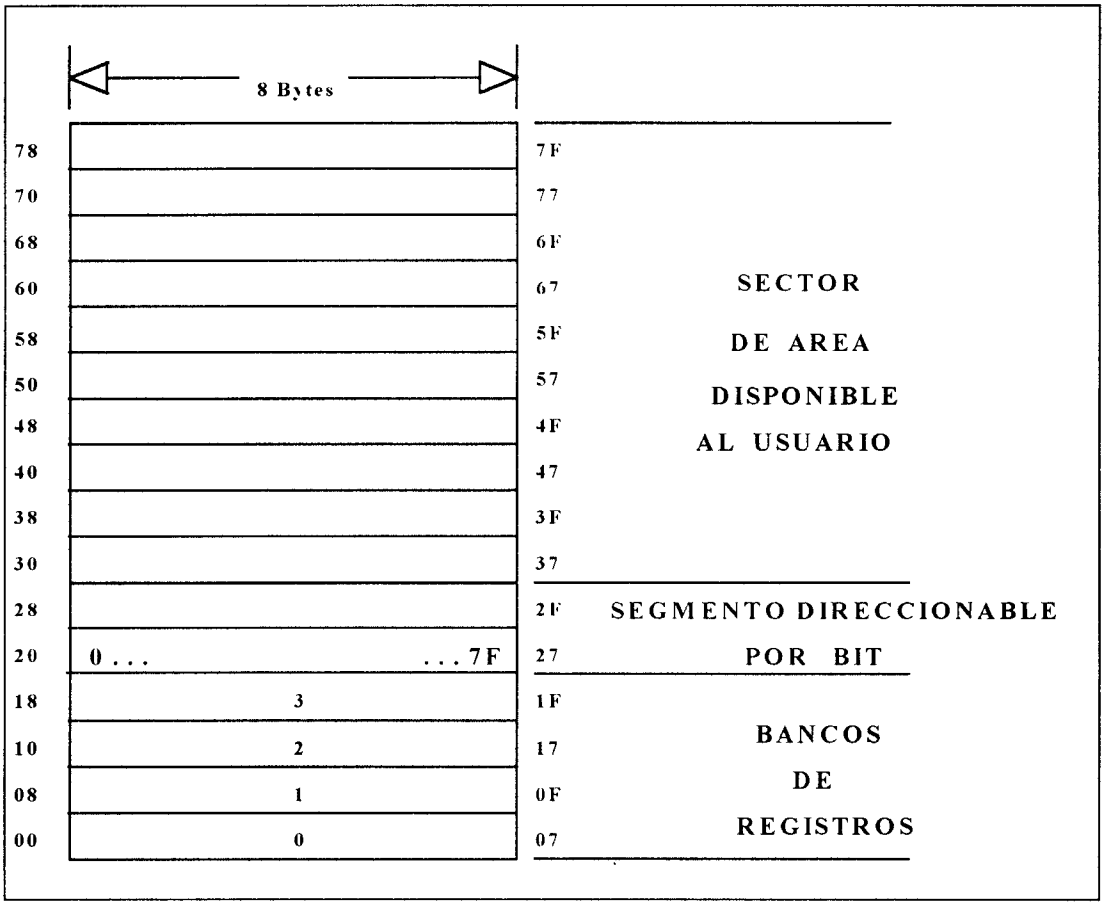


Figura 3.6 128 Bytes de RAM Direccionables Directa e Indirectamente.

• Bancos de Registros 0-3· Se sitúan entre 0 y 1FH (32 bytes). ASM-51 y los dispositivos después de reset se ubican en el banco de registro 0. Para usar los otros bancos de registros, el usuario los debe seleccionar en el software (mirar la Guía de Usuario del MCS-51 Micro Assembler). Cada banco de registro contiene 8 registros de un byte, de 0 a 7.

El Reset inicializa el Stack Pointer en la ubicación 07H y es incrementado una vez para comenzar desde la ubicación 08H que es el primer registro (R0) del segundo banco de registro. Así, para usar más de un banco de registro, el Stack Pointer debería ser inicializado en una ubicación de la RAM donde no se este usando para almacenamiento de datos (ej., la parte más alta de la RAM).

• Área Direccionable por Bit 16 bytes se han asignado para este segmento, 20H-2FH. Cada uno de los 128 bits de este segmento puede ser directamente direccionado (0-7FH).

Los bits pueden ser referidos de dos maneras y ambas son aceptadas por el AMS-51. Una manera es referirse a sus direcciones 0 a 7FH. La otra manera esta con referencia a los bytes 20H a 2FH. Así, los bits 0-7 pueden también ser referenciados como los bits 20 0-20.7, y los bits 8-FH son igual a 21.0-21.7. Cada uno de los 16 bytes en el segmento se pueden direccionar como un byte.

• Sector de Área Disponible: Los Bytes 30H a 7FH están disponibles al usuario como datos de RAM. Sin embargo, si el Stack Pointer ha sido inicializado en esta área, un número suficiente de bytes deben ser dejados libres para prevenir la destrucción de datos del Stack Pointer.

Tabla 3.6. Area de Registro de Funciones Especiales (SFR)

SÍMBOLO	NOMBRE	DIRECCIÓN
* ACC	Acumulador	0E0H
* B	Registro B	0F0H
* PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer (2 Bytes)	
DPL	Byte Bajo	82H
DPH	Byte Alto	83H
* P0	Puerto 0	80H
* P1	Puerto 1	90H
* P2	Puerto 2	0A0H
* P3	Puerto 3	0B0H
* IP	Control de Prioridad de Interrupción	0B8H
* IE	Control de Habilitación de Interrupción	0A8H
TMOD	Modo de Control del Timer/Counter	89H
* TCON	Control del Timer/Counter	88H
*+ T2CON	Control del Timer /Counter 2	0C8H
TH0	Byte Alto del Timer/Counter 0	8CH
TL0	Byte Bajo del Timer/Counter 0	8AH
TH1	Byte Alto del Timer/Counter 1	8DH
TL1	Byte Bajo del Timer/Counter 1	8BH
+ TH2	Byte Alto del Timer/Counter 2	0CDH
+ TL2	Byte Bajo del Timer/Counter 2	0CCH
+ RCAP2H	Byte Alto del Reg. de captura T/C 2	0CBH
+ RCAP2L	Byte Bajo del Reg. de Captura T/C 2	0CAH
* SCON	Control Serial	98H
SBUF	Dato del Buffer Serial	99H
PCON	Control de Potencia (Power)	87H

* = Direccionable por Bit.
+ = Solamente en el 8052.

3.6.5. Registros de Funciones Especiales

La Tabla 3.6 contiene una lista de todos los SFRs y sus direcciones.

Comparando la Tabla 3.6 y la Figura 3.7 se aprecian que todos los SFRs que son byte y bit direccionables están localizados sobre la primera columna del diagrama en la Figura 3.7.

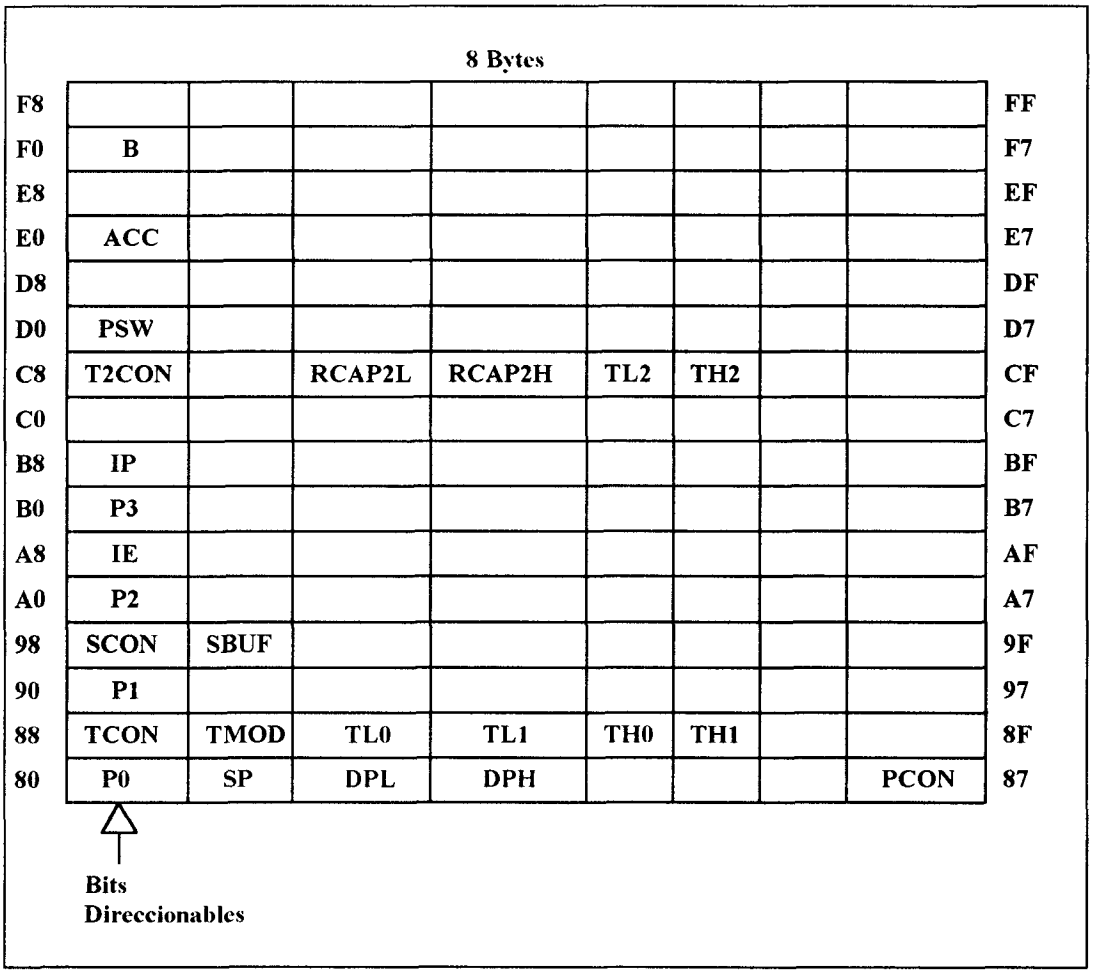


Figura 3.7. Mapa de Memoria de los SFR.

3.6.6. Operación Reset

¿ Que Contienen los SFRs Justo Después del Encendido o de Reset ?

La Tabla 3.7. enumera los contenidos de cada uno de los SFR después de encender o de producir un reset por hardware.

Esos SFRs que tienen asignados bits para diversas funciones se enumeran en esta sección. Una descripción breve de cada bit se da como referencia rápida.

Tabla 3.7. Contenido de los SFRs Después del Reset

Registro	Valor en Binario
*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000, 8052 XX000000
*IE	8051 0XX00000, 8052 0X000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
+RCAP2H	00000000
+RCAP2L	00000000
*SCON	00000000
SBUF	Indeterminado
PCON	HMOS 0XXXXXXX CHMOS 0XXX0000

X = No definido
* = Direccionable por Bit.
+ = Solamente para los 8052.

PSW: Program Status Word (Palabra de Condición de Programa). Direccionable por Bit.

CY	AC	F0	RS1	RS0	OV	-----	P

CY PSW.7 Bandera de Carry.

AC PSW.6 Bandera de Carry Auxiliar.

F0 PSW.5 Bandera 0, disponible al usuario para el propósito general.

RS1 PSW.4 Selector de banco de registro bit 1 (VER NOTA).

RS0 PSW.3 Selector de banco de registro bit 0 (VER NOTA).

OV PSW.2 Bandera de Sobreflujo.

---- PSW.1 No implementado. Reservado para el uso futuro.*

P PSW.0 Bandera de paridad. Es Setiada/Resetiada por Hardware cada ciclo de instrucción para indicar un número par/impar de bits en "1" en el Acumulador.

NOTA:

1. El valor presentado por RS0 y RS1 selecciona el correspondiente banco de registro.

RS1	RS0	Banco de Registro	Dirección
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

* El usuario del software no debería escribir 1s en los bits reservados. Estos bits pueden usarse en futuros productos MCS-51 para añadir aspectos nuevos. En el caso de reset, el valor inactivo del nuevo bit puede ser 0, y su valor activo 1.

PCON: Power Control Register (Registro de Control de Potencia). no Direccionable por Bit.

SMOD	-------	-------	--------	GF1	GF0	PD	IDL

SMOD Double baud rate bit (bit de doble velocidad de transmisión). Si el Timer 1 se

usa a la velocidad de transmisión en baudios producida, y SMOD = 1, la velocidad de transmisión en baudios se dobla cuando el Puerto Serial es usado en modos 1, 2 o 3.

---- No implementado.

---- No implementado.

---- No implementado.

GF1 Bit de bandera de propósito general.

GF0 Bit de bandera de propósito general.

PD Bit de baja potencia. Este bit activa la operación Power Down (baja potencia) en el 80C51BH (Disponible solamente en CHMOS)

IDL Idle Mode bit (bit de Modo Lento). Este bit activa la operación Idle Mode en el 80C51BH. (Disponible solamente en CHMOS).

Si se escriben 1s en IDL y PD a la vez, PD tiene prioridad.

3.6.7. Interrupciones

A fin de usar cualquiera de las interrupciones en el MCS-51, los siguientes tres pasos deben ser seguidos:

1. Coloque el bit EA (permitir todo) del registro IE a 1
2. Habilite el bit correspondiente a la interrupción en el registro IE con 1.
3. Comience la rutina de servicio de la interrupción en la dirección del vector

correspondiente a la interrupción. Ver tabla 3.8

Tabla 3.8. Vectorización de las Interrupciones

Fuente de Interrupción	Dirección del Vector
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H
TF2 & EXF2	002BH

Además, para interrupciones externas, los pines INT0 y INT1 (P3.2 y P3.3) deben colocarse a 1, dependiendo de si la interrupción es activa por nivel o por transición (Flanco), los bits IT0 o IT1 en el registro TCON pueden necesitar ser puestos en 1.

ITx = 0 Activado en nivel.

ITx = 1 Activado en transición (Flanco).

IE: Registro Para Habilitar Interrupciones. Direccionable Por Bit.

Si el bit es 0, la correspondiente interrupción es incapacitada. Si el bit es 1, la correspondiente interrupción se permite.

EA	-----	ET2	ES	ET1	EX1	ET0	EX0

EA IE.7 Inhabilita todas las interrupciones. Si EA=0, ninguna interrupción se reconoce. Si EA=1, cada fuente de interrupción individualmente se permite o se incapacita fijando o limpiando los bits habilitados para este fin.

---- IE.6 No implementado.

ET2 IE.5 Habilita o inhabilita la interrupción de sobreflujo del Timer 2 o interrupción de captura (solamente en el 8052).

ES IE.4 Habilita o inhabilita la interrupción del Puerto Serial.

ET1 IE.3 Habilita o inhabilita la interrupción de sobreflujo del Timer 1.

EX1 IE.2 Habilita o inhabilita la Interrupción Externa 1

ET0 IE.1 Habilita o inhabilita la interrupción de sobreflujo del Timer 0.

EX0 IE.0 Habilita o inhabilita la Interrupción Externa 0.

3.6.7.1. Asignación de la mas alta Prioridad a una o mas Interrupciones

A fin de asignar la mayor prioridad para una interrupción, el bit correspondiente en el registro IP debe colocarse en 1. Recuerde que mientras un servicio de interrupción se cumple, no puede ser interrumpido por uno más bajo o igual nivel de interrupción.

3.6.7.2. La Prioridad Dentro del Nivel

La prioridad dentro del nivel es única para resolver pedidos simultáneos del mismo nivel de prioridad. De mayor a menor, se enumeran a continuación las fuentes de interrupción:

IE0

TF0

IE1

TF1

RI & TI

TF2 O EXF2

IP: Registro de Prioridad de Interrupción. Direccionable Por Bit.

Si el bit es 0, la correspondiente interrupción tiene una prioridad menor y si el bit es 1 la correspondiente interrupción tiene una prioridad más alta.

-----	-----	PT2	PS	PT1	PX1	PT0	PX0

---- IP.7 No implementado.

---- IP.6 No implementado.

PT2 IP.5 Define el nivel de prioridad de la interrupción del Timer 2. (Solamente en el 8052).

PS IP.4 Define el nivel de prioridad de la interrupción del Puerto Serial.

PT1 IP.3 Define el nivel de prioridad de la interrupción del Timer 1.

PX1 IP.2 Define el nivel de prioridad de la Interrupción Externa 1.

PT0 IP.1 Define el nivel de prioridad de la interrupción del Timer 0.

PX0 IP.0 Define el nivel de prioridad de la interrupción Externa 0

TCON: Registro de Control Timer/Contador. Direccionable Por Bit.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 TCON.7 Bandera de sobreflujo del Timer 1. Se fija por hardware cuando el Timer/Contador 1 desborda. Es limpiado por hardware en el momento en que se presenta un servicio de vectores de rutina de interrupción.

TR1 TCON.6 Bit de control de ejecución del Timer 1. Se fija o limpia por software para poner el Timer/Contador 1 en ON/OFF.

TF0 TCON.5 Bandera de sobreflujo del Timer 0. Se fija por hardware cuando el Timer/Contador 0 desborda. Es limpiado por hardware en el momento en que se presenta un servicio de vectores de rutina de interrupción.

TR0 TCON.4 Bit de control de ejecución del Timer 0. Se fija o limpia por software para poner el Timer/Counter 0 en ON/OFF.

IE1 TCON.3 Bandera de borde de la Interrupción Externa 1. Es Setiada por hardware cuando es detectado un cambio (borde) en la interrupción Externa. Es Resetiada por hardware cuando la interrupción es ejecutada.

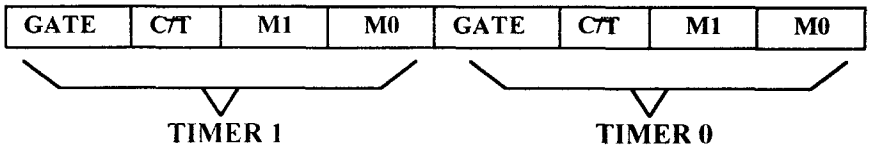
IT1 TCON.2 Bit de control de tipo interrupción 1 Es Setiada/Resetiada por

software para especificar (borde descendente)/(nivel bajo) provocado por la Interrupción Externa.

IE0 TCON.1 Bandera de borde de la Interrupción Externa 0. Es Setiada por hardware cuando es detectado un cambio (borde) en la interrupción Externa. Es Resetida por hardware cuando la interrupción es ejecutada.

IT0 TCON.0 Bit de control de tipo interrupción 0. Es Setiada/Resetiada por software para especificar (borde descendente)/(nivel bajo) provocado por la Interrupción Externa.

TMOD: Registro de Control Del Modo Timer/Contador. no Direccionable Por Bit.



GATE Cuando TRx (en TCON) es puesto y GATE = 1, el TIMER/CONTADORx corre únicamente mientras TRx = 1 (controlado por software).

C/T Selector de Timer o Contador. Se Resetea para la operación del Timer (entrada de reloj interno del sistema). Es Setiada para operar como Contador (entrada desde el Pin de Tx de entrada).

M1 Bit Selector de Modo (NOTA 1).

M0 Bit Selector de Modo. (NOTA 1).

NOTA 1:

M1	M0	Modo de Operación
0	0	0 13-bit de Timer (Compatible con MCS-48)
0	1	1 16-bit de Timer/Contador.
1	0	2 8-bit Timer/Contador Auto-Reajustables.
1	1	3 (Timer 0) TL0 es un Timer/Contador de 8-bit y es controlado por los bits de control del Timer 0 standard. TH0 es un Timer de 8-bit y es controlado por los bits del Timer 1.
1	1	3 (Timer 1) Timer/Contador 1 parado.

3.6.8. Temporizadores y Contadores

3.6.8.1. Estructuración del Timer

Desde la Tabla 3.9 hasta la 3.12 se dan algunos valores para TMOD que pueden ser usados para establecer el Timer 0 en diferentes modos.

Se presume que un único Timer es usado en un momento dado. Si se desea que funcionen los Timers 0 y 1 simultáneamente, en cualquier modo, se debe hacer lo siguiente: se debe realizar una operación lógica 0R entre el valor en TMOD para el Timer 0 y el valor mostrado en TMOD para el Timer 1 (Tablas 3.11 y 3.12).

Por ejemplo, si se desea que corra el Timer 0 en el modo 1 GATE (Control externo), y el Timer 1 en modo 2 CONTADOR, entonces el valor que debe cargarse en TMOD es 69H (09H de la Tabla 3.9 efectuando la operación lógica OR con 60H de la Tabla 3.12). Además, el usuario debe ser conciente que en este punto, no están preparados

los Timers para ser encendidos y debe hacer que en un punto diferente en el programa se fije el bit TRx (en TCON) a 1.

3.6.8.2. Timer/Contador 0

Como un Timer:

Tabla 3.9 Características del Timer 0

MODO	FUNCION DEL TIMER 0	TMOD	
		CONTROL INTERNO (NOTA 1)	CONTROL EXTERNO (NOTA 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Recargables	02H	0AH
3	Dos Timers de 8-bit	03H	0BH

Como un Contador :

Tabla 3.10 Características del Contador 0

MODO	FUNCION DEL CONTADOR 0	TMOD	
		CONTROL INTERNO (NOTA 1)	CONTROL EXTERNO (NOTA 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Recargables	06H	0EH
3	Un Contador de 8-bit	07H	0FH

NOTAS:

1. Los Timer se cambian de ON/OFF colocando o limpiando el bit TR0 en el software.

2. Los Timer se cambian de ON/OFF en la transición de 1 a 0 en el INT0 (P3.2) cuando TR0 =1 (controlado por hardware).

3.6.8.3. Timer/Contador 1

Como un Timer:

Tabla 3.11 Características del Timer 1

MODO	FUNCION DEL TIMER 1	TMOD	
		CONTROL INTERNO (NOTA 1)	CONTROL EXTERNO (NOTA 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Recargables	20H	A0H
3	No Ejecutable	30H	B0H

Como un Contador:

Tabla 3.12 Características del Contador 1

MODO	FUNCION DEL CONTADOR 1	TMOD	
		CONTROL INTERNO (NOTA 1)	CONTROL EXTERNO (NOTA 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Recargables	60H	E0H
3	No Habilitado	----	----

NOTAS:

1.Los Timer se cambian de ON/OFF colocando o limpiando el bit TR1 en el software.

2.Los Timer se cambian de ON/OFF en la transición de 1 a 0 en el INT1 (P3.3) cuando TR1 =1 (controlado por hardware).

SCON: Registro de Control Del Puerto Serial. Direccionable Por Bit.

SMOD	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON 7 Modo especifico del Puerto Serial. (NOTA 1.)

SM1 SCON 6 Modo especifico del Puerto Serial. (NOTA 1.)

SM2 SCON.5 El multiproceso habilita aspectos de comunicación en modos 2 & 3. En el modo 2 o 3, si SM2 es puesto en 1 entonces RI no puede ser activado si el noveno bit de dato (RB8) es 0. En el modo 1, si SM2=1 entonces RI no puede ser activado si un bit valido de stop no se recibe. (Mirar la Tabla 3.13.)

REN SCON.4 Se fija/limpia por software para habilitar/deshabilitar la recepción.

TB8 SCON.3 Indica el noveno bit que se transmite en modo 2 & 3. Fijado o limpiado por software.

RB8 SCON.2 En modo 2 & 3, denota el noveno bit de dato que se recibe. En modo 1, sí SM2=0, RB8 es el bit de stop que se recibe. En modo 0, RB8 no es usado.

TI SCON.1 Bandera de interrumpir transmisión. Fijada por hardware al final del octavo bit de tiempo en modo 0, o en el comienzo de el bit de stop en los otros modos. Debe ser limpiado por software.

RI SCON.0 Bandera de interrumpir recepción. Fijada por hardware al final del octavo bit de tiempo en modo 0, o en el intermedio del bit de tiempo de stop en los otros modos (exceptuando SM2). Debe ser limpiado por software.

NOTA 1 :

SM0	SM1	Mode	Descripcion	Baud Rate
0	0	0	SHIFT-REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 o Fosc./32
1	1	3	9-Bit UART	Variable

3.6.9. Comunicaciones

3.6.9.1. Estructuración del Puerto Serial

Tabla 3.13. Puerto Serial

MODO	SCON	VARIACION DE SM2
0	10H	Single Procesor
1	50H	Environment
2	90H	(SM2 = 0)
3	D0H	
0	NA	Multiprocessor
1	70	Environment
2	B0	(SM2 = 1)
3	F0	

3.6.9.2. Generacion de Velocidad de Transmision en Baudios

Puerto Serial en Modo 0:

El modo 0 tiene una velocidad de transmisión en baudios fija que es 1/12 de la frecuencia del oscilador Al funcionar el puerto serial en este modo ninguno de los Timer/Contadores necesita ser establecido. Únicamente el registro SCON necesita ser definido.

$$Band\ Rate = \frac{Frec\ Osc}{12}$$

Puerto Serial en Modo 1:

El modo 1 tiene una velocidad de transmisión en baudios variable. La velocidad de transmisión en baudios puede ser generada por cualquiera de los dos Timers (Timer 2 solamente en el 8052).

3.6.9.3. Forma de Usar el Timer/Contador 1 Para Generar Velocidad de Transmision en Baudios

Para este fin, el Timer 1 se usa en el modo 2 (Auto-Reajustable). Refiérase a la sección de estructuración del Timer de este capitulo.

$$Baud\ Rate = \frac{K * Frec.\ Osc.}{32 * 12 * [256 - (TH1)]}$$

Si SMOD = 0, entonces K = 1

Si SMOD = 1, entonces K = 2, (SMOD está en el Registro PCON).

La mayoría del tiempo el usuario conoce la velocidad de transmisión en baudios y necesita saber el valor reajustable para TH1. Por lo tanto, la ecuación para calcular TH1 puede escribirse como:

$$TH1 = 256 - \frac{K * Frec.\ Osc.}{384 * Baud\ Rate}$$

TH1 debe ser un valor entero. Redondeando por encima TH1 al entero más cercano puede no producirse la velocidad de transmisión en baudios deseada. En este caso, el usuario puede que tenga que escoger otra frecuencia de cristal. Dado que el registro PCON no es direccionable por bit, una manera para colocar el bit es realizar una operación lógica OR con el registro PCON. (ej, ORL PCON,#80H). La dirección de PCON es 87H.

Puerto Serial en Modo 2

La velocidad de transmisión en baudios es fija en este modo y es 1/32 o 1/64 de la frecuencia de oscilación que depende del valor del bit SMOD en el registro PCON.
En este modo ninguno de los Timers se usa y el reloj viene desde la fase 2 del reloj interno.

SMOD = 1, Baud Rate = 1/32 Frec. Osc.

SMOD = 0, Baud Rate = 1/64 Frec. Osc.

Para fijar el bit SMOD: ORL PCON,#80H. La dirección de PCON es 87H.

Puerto Serial en Modo 3:

La velocidad de transmisión en baudios en el modo 3 es variable y se establece

exactamente de la misma forma que en el modo 1.

3.6.10. Set de Instrucciones para el MCS-51

Tabla 3.14. Estado de las banderas

Instrucciones que Afectan el Estado de las Banderas. (*)							
Instrucción	**Bandera**			**Instrucción**	**Bandera**		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						
(*) Note qué al hacer operaciones sobre los SFR's, se pueden ver afectados los estados de las banderas.							

Tabla 3.15. Modos de direccionamiento

Notas sobre los Modos de Dirreccionamiento	
Rn	Registro R7-R0 del Banco de Registros a usar.
directo	Dirección de ubicación de 8-bit de datos internos. Estos podrían ser Datos Internos de RAM localizados (0-127) o un SFRs [ej., Puerto I/O, registro de control, registro de condición (status), etc. (128-255)].
@Ri	8-bit de datos internos de la RAM localizados entre (0-255) direccionados indirectamente mediante los registros R1 o R0.
#dato	8-bit constantes incluidos en la instrucción.
#dato16	16-bit constantes incluidos en la instrucción
addr16	16-bit de destino de dirección. Usado por LCALL & LJMP. Un salto puede ser a donde se quiera dentro de los 64K bytes del espacio de dirección de Memoria de Programa
addr11	11-bit de destino de dirección. Usado por ACALL & AJMP. El salto debe estar dentro de los 2K bytes de la pagina de Memoria de Programa como el primer byte de la siguiente instrucción.
rel	8-bit signados (complemento a dos) compensan el byte. Usado por SJMP y todos los saltos condicionales. El rango es -128 a +127 bytes relativos a la primera de las instrucciones siguientes.
bit	Bit directo direccionado a dato interno de la RAM o a Registro de Funciones Especiales.
*	Operaciones nuevas no existentes en el 8048AH/8049AH.

Tabla 3.16a. Set de Instrucciones

Resumen del Set de Instrucciones de 8051			
Mnemónico	**Descripción**	**Byte**	**Periodos de Oscilación**
OPERACIONES ARITMÉTICAS			
ADD A.Rn	Sume el registro a el ACC	1	12
ADD A.directo	Sume byte directo a el ACC	2	12
ADD A.@Ri	Suma RAM indirecta a el ACC	1	12
ADD A.#dato	Suma el dato inmediato a el ACC	2	12
ADDC A.Rn	Suma con Carry el registro a el ACC	1	12
ADDC A.directo	Suma con Carry byte directo a el ACC	2	12
ADDC A.@Ri	Suma con Carry indirectamente la RAM a el ACC	1	12
ADDC A.#dato	Suma con Carry el dato inmediato a el ACC	2	12
SUBB A.Rn	Resta el registro con el ACC pidiendo prestado	1	12
SUBB A.directo	Resta directamente el byte con el ACC pidiendo prestado	2	12
SUBB A.@Ri	Resta RAM indirecta con el ACC pidiendo prestado	1	12
SUBB A.#dato	Resta el dato inmediato con el ACC pidiendo prestado	2	12
INC A	Incrementa el ACC	1	12
INC Rn	Incrementa registro	1	12
INC directo	Incrementa byte directo	2	12
INC @Ri	Incrementa RAM indirecta	1	12
DEC A	Decrementa el ACC	1	12
DEC Rn	Decrementa registro	1	1
DEC directo	Decrementa byte directo	2	12
DEC @Ri	Decrementa RAM indirecta	1	12
INC DPTR	Incrementa el Data Pointer	1	24

Tabla 3.16b Set de Instrucciones

Mnemónico	Descripción	Byte	Periodos de Oscilación
OPERACIONES ARITMÉTICAS (continuación)			
MUL AB	Multiplica ACC por el registro B	1	48
DIV AB	Divide el ACC en el registro B	1	48
DA A	Ajuste decimal en ACC		
OPERACIONES LÓGICAS			
ANL A,Rn	AND de registro y el ACC	1	12
ANL A,directo	AND de byte directo y el ACC	2	12
ANL A,@Ri	AND de RAM indirecta y el ACC	1	12
ANL A,#dato	AND de dato inmediato y ACC	2	12
ANL directo,A	AND de ACC y byte directo	2	12
ANL directo,#dato	AND de dato inmediato y byte directo	3	24
ORL A,Rn	OR de registro y el ACC	1	12
ORL A,directo	OR de byte directo y el ACC	2	12
ORL A,@Ri	OR de RAM indirecta y ACC	1	12
ORL A,#dato	OR de dato inmediato y ACC	2	12
ORL directo,A	OR de ACC y byte directo	2	12
ORL directo,#dato	OR de dato inmediato y byte directo	3	24
XRL A,Rn	OR-Exclusivo de registro y el ACC	1	12
XRL A,directo	OR-Exclusivo de byte directo y el ACC	2	12
XRL A,@Ri	OR-Exclusivo de RAM indirecta y el ACC	1	12
XRL A,#dato	OR-Exclusivo de dato inmediato y el ACC	2	12
XRL directo,A	OR-Exclusivo de ACC y el byte directo	2	
XRL directo,#dato	OR-Exclusivo de dato inmediato y el byte directo	3	
CLR A	Borra el ACC	1	12

Tabla 3.16c Set de Instrucciones

Mnemónico	Descripción	Byte	Periodos de Oscilación
OPERACIONES LÓGICAS (continuación)			
CPL A	Complementa el ACC	1	12
RL A	Rota a la izquierda el ACC	1	12
RLC A	Rota a la izquierda el ACC a través del Carry	1	12
RR A	Rota a la derecha el ACC	1	12
RRC A	Rota a la derecha el ACC a través del Carry	1	12
SWAP A	Intercambia nibbles dentro de el ACC	1	12
TRASLADAR DATO			
MOV A.Rn	Mueve el registro a el ACC	1	12
MOV A.directo	Mueve el byte directo a el ACC	2	12
MOV A.@Ri	Mueve RAM indirecta a el ACC	1	12
MOV A.#dato	Mueve dato inmediato a el ACC	2	12
MOV Rn.A	Mueve el ACC a el registro	1	12
MOV Rn.directo	Mueve byte directo a el registro	2	24
MOV Rn.#dato	Mueve dato inmediato a el registro	2	12
MOV directo.A	Mueve el ACC a el byte directo	2	12
MOV direct.Rn	Mueve registro a byte directo	2	24
MOV direct.direct	Mueve byte directo a byte directo	3	24
MOV directo.@Ri	Mueve RAM indirecta a byte directo	2	24
MOV directo.#dato	Mueve dato inmediato a byte directo	3	24
MOV @Ri.A	Mueve el ACC a RAM indirecta	1	12
MOV @Ri.directo	Mueve byte directo a RAM indirecta	2	24
MOV @Ri.#dato	Mueve dato inmediato a RAM indirecta	2	12

Tabla 3.16d Set de Instrucciones

Mnemónico	Descripción	Byte	Periodos de Oscilación
TRASLADAR DATO (continuación)			
MOVC DPTR,#dato16	Carga Data Pointer con una constante de 16-bit	3	24
MOVC A,@A+DPTR	Mueve el código del byte relativo de el DPTR a el ACC	1	24
MOVC A,@A+PC	Mueve el código del byte relativo de PC a el ACC	1	24
MOVX A,@Ri	Mueve RAM externa (8-bit addr) a el ACC	1	24
MOVX A,@DPTR	Mueve RAM externa (16-bit addr) a el ACC	1	24
MOVX @Ri,A	Mueve ACC a RAM externa (8-bit addr)	1	24
MOVX @DPTR,A	Mueve ACC a RAM externa (16-bit addr)	1	24
PUSH directo	Mete byte directo dentro de el Stack	2	24
POP directo	Saca byte directo de el Stack	2	24
XCH A,Rn	Intercambia registro con el ACC	1	12
XCH A,directo	Intercambia byte directo con el ACC	2	12
XCH A,@Ri	Intercambia RAM indirecta con el ACC	1	12
XCHD A,@Ri	Intercambia Dígito de menor orden de RAM indirecta con el ACC	1	12
MANIPULACIÓN DE VARIABLES BOOLEANAS			
CLR C	Borrar Carry	1	12
CLR bit	Borrar bit directo	2	12
SETB C	Cargar Carry con 1	1	12
SETB bit	Cargar bit directo con 1	2	12
CPL C	Complementa ACC	1	12
CPL bit	Complementa bit directo	2	12
ANL C,bit	AND de bit directo a Carry	2	24
ANL C,/bit	AND del complemento de bit directo a Carry	2	24

Tabla 3.16e Set de Instrucciones

Mnemónico	Descripción	Byte	Periodos de Oscilación
MANIPULACIÓN DE VARIABLES BOOLEANAS (continuación)			
ORL C.bit	OR de bit directo y Carry	2	24
ORL C./bit	OR de complemento de bit directo y Carry	2	24
MOV C.bit	Mueve bit directo a Carry	2	12
MOV bit.C	Mueve Carry a bit directo	2	24
JC rel	Salta si hay Carry	2	24
JNC rel	Salta si no hay Carry	2	24
JB bit.rel	Salta si hay puesto bit directo	3	24
JNB bit.rel	Salta si no hay puesto bit directo	3	24
JBC bit.rel	Salta si hay puesto bit directo & borra bit	3	24
PROGRAMAR SALTOS			
ACALL addr11	Llamada Absoluta a subrutina	2	24
LCALL addr16	Llamada Larga a subrutina	3	24
RET	Retorne de Subrutina	1	24
RETI	Retorne de Interrupción	1	24
AJMP addr11	Salto Absoluto	2	24
LJMP addr16	Salto Largo	3	24
SJMP rel	Salto Corto (add relativa)	2	24
JMP @A+DPTR	Salto relativo indirecto a el DPTR	1	24
JZ rel	Salte si el ACC es cero	2	24
JNZ rel	Salte si el ACC no es cero	2	24
CJNE A.directo.rel	Compare byte directo con ACC y Salte si no son iguales	3	24
CJNE A.#dato.rel	Compare dato inmediato con ACC y Salte si no son iguales	3	24
CJNE Rn.#dato.rel	Compare dato inmediato con registro y Salte si no son iguales	3	24

Tabla 3.16f Set de Instrucciones

Mnemónico	Descripción	Byte	Periodos de Oscilación
PROGRAMAR SALTOS (continuación)			
CJNE @Ri,#dato,rel	Compare dato inmediato con RAM interna y Salte si no son iguales	3	24
DJNZ Rn,rel	Decremente registro y salte si no es cero.	2	24
DJNZ directo,rel	Decremente byte directo y salte si no es cero.	3	24
NOP	No Operación	1	12

3.6.10.1. Definición de las Instrucciones

- **ACALL addr 11**

Función: Llamada Absoluta.

Descripción: ACALL incondicionalmente llama una subrutina ubicada en la dirección indicada. La instrucción incrementa el PC dos veces para obtener la dirección de la instrucción siguiente, entonces empuja el resultado de los 16 bit en el Stack (los primeros bytes de menor orden) e incrementa el Stack Pointer dos veces. La dirección de destino es obtenida por concatenaciones sucesivas de los cinco bits de mayor peso del PC incrementaron los bits 7-5 de opcode, y el segundo byte de la instrucción. La subrutina llamada debe comenzar por lo tanto dentro del mismo bloque de 2K de la memoria de programa y el primer byte de la instrucción siguiente a ACALL. Ninguna de las banderas es afectada.

Ejemplo: Inicialmente el SP es igual a 07H. La etiqueta (label) "SUBRTN" está ubicada en 0345H de la memoria de programa Después de ejecutar la instrucción **ACALL SUBRTN,** en la posición 0123H, el SP contendrá 09H, las localizaciones de la RAM interna 08H y 09H contendrán 25H y 01H, respectivamente, y el PC contendrán 0345H.

Bytes. 2

Ciclos: 2

Operación: ACALL

(PC) $\Leftarrow$ (PC) + 2

(SP) $\Leftarrow$ (SP) + 1

((SP)) $\Leftarrow$ (PC7-0) + 2

(SP) $\Leftarrow$ (SP) + 1

((SP)) $\Leftarrow$ (PC15-8)

(PC10-0) $\Leftarrow$ dirección de página

- **ADD A, <src-byte>**

Función: Adiciona.

Descripción: ADD adiciona el byte de la variable al Acumulador, dejando el resultado

en el Acumulador. Las banderas de Carry y carry-auxiliar son setiadas, si resulta Carry del bit 7 o el bit 3 respectivamente, y resetiado de otra manera. Cuando se adicionan enteros no signados, la bandera de carry indica que un sobreflujo ha ocurrido.
OV es setiada si hay un carry fuera del bit 6 pero no fuera del bit 7, o un carry fuera del bit 7 pero no en el bit 6, de otra manera OV es resetiado. Cuando se adicionan enteros signados, OV indica que un número negativo se produjo como la suma de dos operandos positivos, o una suma positiva de dos operandos negativos.

Cuatro fuentes de aplicación permiten los modos de direccionamiento: registro, directo, registro-indirecto, o inmediato

Ejemplo: El Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B).

La instrucción **ADD A,R0,** retorna 6DH (01101101B) en el acumulador con la bandera de AC resetiada y las otras dos banderas (la de carry y la de OV) puestas en 1.

ADD A,Rn

Bytes: 1

Ciclos: 1

Operación: ADD

(A) $\Leftarrow$ (A) + (Rn)

ADD A,directo

Bytes: 2

Ciclos: 1

Operación: ADD

(A) $\Leftarrow$ (A) + (directo)

ADD A,@ Ri

Bytes: 1

Ciclos: 1

Operación: ADD

(A) $\Leftarrow$ (A) + ((Ri))

- **ADDC A, <src-byte>**

Función: Suma con Carry

Descripción: ADDC simultáneamente agrega el byte indicado en la variable, con los contenidos de la bandera de Carry y el acumulador, dejando el resultado en el acumulador. Las banderas de Carry y Carry-auxiliar son setiadas, si resulta Carry del bit 7 o el bit 3 respectivamente, y resetiadas de otra manera. Cuando se adicionan enteros no signados, la bandera de carry indica que un sobreflujo ha ocurrido.

OV es setiada si hay un carry fuera del bit 6 pero no en el bit 7, o un carry fuera del bit 7 pero no en el bit 6; de otra manera OV es reseiada. Cuando se suman enteros signados, OV indica que un número negativo se produjo como la suma de dos operandos positivos, o una suma positiva de dos operandos negativos.

Cuatro fuentes de aplicación permiten los modos de direccionamiento: registro, directo, registro-indirecto, o inmediato.

Ejemplo: El Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B) con la bandera de carry puesta. La instrucción **ADDC A,R0,** retorna 6EH (01101110B) en el Acumulador con la bandera de AC resetiada y las otras dos

banderas (la de carry y la de OV) puestas en 1.

ADDC A,Rn

Bytes: 1

Ciclos: 1

Operación: ADDC

(A) ⇐ (C) + (Rn)

ADDC A,directo

Bytes: 2

Ciclos: 1

Operación: ADDC

(A) ⇐ (A) + (C) + (directo)

ADDC A,@ Ri

Bytes: 1

Ciclos: 1

Operación: ADDC

(A) ⇐ (A) + (C) + ((Ri))

ADDC A,#dato

Bytes: 2

Ciclos: 1

Operación: ADDC

(A) $\Leftarrow$ (A) + (C) + #dato

- **AJMP addr 11**

Función: Salto absoluto.

Descripción: AJMP programa la ejecución de los saltos a la dirección indicada, que se forma en el tiempo de ejecución por la concatenación de los cinco bits de mayor orden del PC (después de incrementar el PC dos veces), los bits 7-5 de opcode, y el segundo byte de la instrucción. El destino debe por lo tanto estar dentro del mismo bloque de 2K de la memoria de programa y el primer byte de la instrucción siguiente a AJMP.

Ejemplo: La etiqueta (label) "JMPADR" está en la ubicación 0123H de la memoria de programa. La instrucción **AJMP JMPADR,** está en la ubicación 0345H y carga el PC con 0123H.

Bytes: 2

Ciclos. 2

Operación: AJMP

(PC) $\Leftarrow$ (PC) + 2

(PC10-0) $\Leftarrow$ dirección de página

- **ANL** **<dest-byte>, <src-byte>**

Función: Función lógica AND entre variables de byte.

Descripción: ANL desempeña la operación lógica discreta AND (Y) entre las variables indicadas y el resultado lo almacena en la variable de destino Ninguna bandera es afectada. Los dos operandos permiten seis combinaciones de modo de direccionamiento. Cuando el destino es el Acumulador, la fuente puede usar direccionamiento directo, registro, registro-indirecto, o inmediato; Cuando el destino es una dirección directa, la fuente puede ser el Acumulador o datos inmediatos.

NOTA: Cuando ésta instrucción se usa para modificar la salida de un puerto, el valor usado como dato original del puerto se lee desde el latch de datos de salida, no de los pines de entrada.

Ejemplo: Si el Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 55H (01010101B) entonces la instrucción **ANL A,R0,** entrega 41H (01000001B) en el Acumulador. Cuando el destino es un byte directamente direccionado, ésta instrucción borra combinaciones de bits en cualquier ubicación de la RAM o registro del

hardware. El byte de mascara determina los tipos de bits que deben ser borrados contenidos en una constante que esta contenida en la instrucción o en un valor computado en el Acumulador en el tiempo de ejecución. La instrucción

ANL P1,#01110011B, resetea los bits 7, 3 y 2 de salida del Puerto 1.

ANL A,Rn

Bytes: 1

Ciclos: 1

Operación: ANL

(A) $\Leftarrow$ (A) $\wedge$ (Rn)

ANL A,directo

Bytes: 2

Ciclos 1

Operación: ANL

(A) $\Leftarrow$ (A) $\wedge$ (directo)

ANL A, @Ri

Bytes: 1

Ciclos: 1

Operación: ANL

(A) $\Leftarrow$ (A) $\wedge$ ((Ri))

ANL A,dato

Bytes: 2

Ciclos: 1

Operación: ANL

(A) $\Leftarrow$ (A) Λ #dato

ANL directo,A

Bytes: 2

Ciclos: 1

Operación: ANL

(directo) $\Leftarrow$ (directo) Λ (A)

ANL directo,#dato

Bytes: 3

Ciclos: 2

Operación: ANL

(directo) $\Leftarrow$ (directo) Λ #dato

- **ANL C, <src-bit>**

Función: Operación lógica AND para variable bit.

Descripción: Si el valor Booleano del bit de fuente es un 0 lógico entonces se borra la bandera de carry; de otra manera la bandera de carry se mantiene en su estado actual. Un slash ("/") precediendo el operando en el lenguaje Assembly indica que el complemento lógico del bit direccionado es usado como el valor de fuente, pero el mismo bit fuente no es afectado. Ninguna otra bandera se afecta. Solamente el direccionamiento directo se permite para el operando de la fuente.

Ejemplo: Pone la bandera de carry si, y solo si, P1.0 = 1, ACC.7 = 1, y OV = 0:

```
MOV  C,P1.0  ; CARGA EL CARRY CON EL ESTADO DEL PIN DE ENTRADA
ANL  C,ACC 7 , (CARRY) AND (ACOMULADOR.BIT7)
ANL  C,/OV ; (CARRY) AND (INVERSO DE LA BANDERA DE OVERFLOW).
```

ANL C,bit

Bytes: 2

Ciclos: 2

Operación: ANL

(C) $\Leftarrow$ (C) Λ (bit)

ANL C./bit

Bytes: 2

Ciclos: 2

Operación: ANL

(C) $\Leftarrow$ (C) Λ] (bit)

- **CJNE < dest-byte>, <src-byte>, rel**

Función: Compare y salte si no es igual.

Descripción: CJNE compara las magnitudes de los primeros dos operandos, y salta si sus valores no son iguales. El destino del salto se computa sumando el desplazamiento relativo signado en el último byte de instrucción en el PC, después de incrementar el PC al comienzo de la próxima instrucción. La bandera de carry es setiada si el valor entero no signado del <dest-byte> es menor que el valor entero no signado de <src-byte>; de otra manera es borrada. Ninguno de los dos operandos es afectado.

Los primeros dos operandos permiten cuatro formas de combinar los modos de direccionamiento; el Acumulador puede ser comparado con cualquier byte

directamente direccionado o dato inmediato, o cualquier ubicación indirecta de la RAM; o trabajando como registro, puede ser comparado con una constante inmediata.

Ejemplo: El Acumulador contiene 34H. El registro 7 contiene 56H. La primera instrucción en la primera secuencia,

```
CJNE    R7,#60H,NOT_EQ
;                  . . .  . . .                    ; R7 = 60H
NOT_EQ             JC     RE      Q_LOW            ; Si R7 < 60H
;                  . . .  . . .                    ; R7 > 60H
```

setea la bandera de carry y salta hasta la instrucción en la etiqueta NOT_EQ. Esta instrucción determina si R7 es mayor o menor a 60H.

Si los datos que están presentes en el Puerto 1 son también 34H, entonces la instrucción **WAIT: CJNE A,P1,WAIT,** resetea la bandera de carry y continua con la siguiente instrucción en la secuencia, ya que el Acumulador iguala a los datos que se leen de P1 (Si algún otro valor se registra en P1, el programa entra en un loop hasta que el dato en P1 cambie a 34H).

CJNE A,directo,rel

Bytes: 3

Ciclos: 2

Operación: (PC) ⇐ (PC) + 3

Si (A) < > (directo)

entonces

(PC) ⇐ (PC) + offset relativo

Si (A) < (directo)

entonces

(C) ⇐ 1

Si no

(C) ⇐ 0

CJNE A,#dato,rel

Bytes: 3

Ciclos: 2

Operación: (PC) ⇐ (PC) + 3

Si (A) < > dato

entonces

(PC) ⇐ (PC) + offset relativo

Si (A) < dato

entonces

(C) ⇐ 1

Si no

(C) ⇐ 0

CJNE Rn,#dato,rel

Bytes: 3

Ciclos: 2

Operación: (PC) ⇐ (PC) + 3

Si (Rn) < > dato

entonces

(PC) ⇐ (PC) + offset relativo

Si (Rn) < dato

entonces

(C) ⇐ 1

Si no

(C) ⇐ 0

CJNE @Ri,#dato,rel

Bytes: 3

Ciclos: 2

Operación: (PC) ⇐ (PC) + 3

Si ((Ri)) < > dato

entonces

(PC) ⇐ (PC) + offset relativo

Si ((Ri)) < dato

entonces

(C) ⇐ 1

Si no

(C) ⇐ 0

- **CLR A**

Función: Borra el Acumulador.

Descripción: El Acumulador es borrado (todo el conjunto de bits son puestos en 0). Ninguna de las banderas es afectada.

Ejemplo. El Acumulador contiene 5CH (01011100B). La instrucción **CLR A,** retorna el Acumulador con el valor 00H (00000000B).

Bytes: 1

Ciclos. 1

Operación: CLR

(A) ⇐ 0

- **CLR bit**

Función: Borra un bit.

Descripción: El bit indicado es borrado. Ninguna otra bandera se afecta. CLR puede operar sobre la bandera de carry o cualquier bit direccionable directamente.

Ejemplo: El Puerto 1 anteriormente se ha escrito con 5DH (01011101B). La instrucción **CLR P1.2,** retorna el Puerto 1 en 59H (01011001B).

CLR C

Bytes: 1

Ciclos: 1

Operación: CLR

(C) $\Leftarrow$ 0

CLR bit

Bytes. 2

Ciclos: 1

Operación: CLR

(bit) $\Leftarrow$ 0

- **CPL A**

Función: Complementa el Acumulador.

Descripción: Cada bit del Acumulador es complementado lógicamente (complemento a 1). Los bits que anteriormente contenían 1 son cambiados por 0 y visceversa.

Ninguna de las banderas es afectada.

Ejemplo: El Acumulador contiene 5CH (01011100B). La instrucción **CPL A,** retorna el Acumulador cargado con 0A3H (10100011B).

Bytes: 1

Ciclos: 1

Operación: CPL

(A) $\Leftarrow$](A) ("]" indica complemento a).

- **CPL bit**

Función: Complemento de bit.

Descripción. La variable bit especificada se complementa. Un bit que ha sido un uno se cambia a cero y visceversa. Ninguna de las banderas es afectada. CPL puede operar sobre la bandera de carry o cualquier bit direccionable directamente.

NOTA: Cuando esta instrucción es usada para modificar un pin de salida, el valor usado como datos originales se lee desde el latch de datos de salida, no de los pines de entrada.

Ejemplo: El Puerto 1 anteriormente se ha escrito con 5DH (01011101B). La secuencia de la insttrucción,

CPL P1.1

CPL P1.2

retorna el Puerto 1 con 5BH (01011011B).

CPL C

Bytes: 1

Ciclos. 1

Operación: CPL

(C) ⇐] (C)

CPL bit

Bytes: 2

Ciclos: 1

Operación: CPL

(bit) ⇐] (bit)

- **DA A**

Función: Ajuste decimal en el Acumulador para la Adición.

Descripción: DA A ajusta el valor de los ocho bits en el Acumulador que resultan de la adición anterior de dos variables, produciendo dos digitos de cuatro bits. Cualquier instrucción ADD o ADDC puede ser usada para desempeñar la adición.

Si los bits 3-0 en el Acumulador son mayor de nueve (xxxx1010-xxxx1111), o si la bandera de AC es uno, se adiciona 6 a el Acumulador produciendo el digito BCD apropiado en el nibble de menor orden. Esta adición interna setea la bandera de carry sí resulta carry de los cuatro bit de menor orden, propagándose hacia todos los cuatro

bit de mayor orden, pero esto no borraría la bandera de carry.

Si la bandera de carry es puesta ahora, o si los cuatro bits de mayor orden exceden a nueve (1010xxxx-1111xxxx), estos bits de mayor orden son incrementado por seis, produciendo el digito BCD apropiado en el nibble de mayor orden. Nuevamente, esto setea la bandera de carry sí los bits de mayor orden sacan carry, pero no borrarían el carry. Así, la bandera de carry indica si la suma de las dos variables BCD originales es mayor a 100, permitiendo la suma múltiple decimal de precisión. OV no es afectado.

Todo esto ocurre durante el único ciclo de instrucción Esencialmente, esta instrucción desempeña la conversión decimal agregando 00H, 60H, 66H al Acumulador, dependiendo de la condicion inicial de el Acumulador y las condiciones de PSW.

Nota: DA A simplemente no puede convertir un número hexadecimal en el Acumulador a notación BCD, ni DA A se aplica a la substracción decimal.

Ejemplo: El Acumulador contiene el valor 56H (01010110B) que representa el digito BCD empaquetado del número decimal 56 El registro 3 contiene el valor 67H (01100111B) que representa el digito BCD empaquetado, del número decimal 67. La bandera de carry es setiada La secuencia de instrucciones,

```
ADDC    A,R3
DA      A
```

lo primero que se hace es adicionar un complemento a dos binario estandar, resultando el valor 0BEH (10111110B) en el Acumulador. Las banderas de Carry y Carry-auxiliar se borran.

La instrucción de Ajuste Decimal altera entonces el Acumulador al valor 24H (00100100B), indicando los digitos BCD empaquetados del número decimal 24, los dos digitos de menor orden de la suma decimal de 56, 67, y 1 es 124.

Los BCD variables pueden ser incrementados o decrementados agregando 01H o 99H. Si el Acumulador inicialmente contiene 30H (representando los digitos a 30 decimal), entonces la secuencia de instrucción,

```
ADD     A,#99
DA      A
```

retorna setiado el carry, y 29H en el Acumulador, de 30 + 99 = 129. El byte de menor orden de la suma puede ser interpretado como 30 - 1 = 29.

Bytes: 1

Ciclos: 1

Operación: DA

El contenido del Acumulador es BCD

Si [[(A3-0) > 9] V [(AC) = 1]]

Entonces

(A3-0) ⇐ (A3-0) + 6

Y Si [[(A7-4) > 9] V [(C) = 1]]

Entonces

(A7-4) ⇐ (A7-4) + 6

- **DEC byte**

Función: Decrementa.

Descripción: La variable indicada es decrementada en 1. Un valor original de 00H llevará a 0FFH. Ninguna de las banderas se afecta Cuatro operadores permiten los modos de direccionamiento Acumulador, registro, directo, registro-indirecto.

NOTA: Cuando esta instrucción se usa para modificar una salida de Puerto, el valor usado como dato original del Puerto se lee desde el latch de datos de salida, no de los pines de entrada

Ejemplo: El registro 0 contiene 7FH (01111111B). Las posiciones 7EH y 7FH de la

RAM interna contienen 00H y 40H, respectivamente. La secuencia de instrucción,

DEC @R0

DEC R0

DEC @R0

retornará el registro 0 con 7EH y las posiciones 7EH y 7FH con 0FFH y 3FH.

DEC A

Byte: 1

Ciclos: 1

Operación: DEC

(A) ⇐ (A) - 1

DEC Rn

Byte: 1

Ciclos: 1

Operación: DEC

(Rn) ⇐ (Rn) - 1

DEC directo

Byte: 2

Ciclos: 1

Operación: DEC

(directo) ⇐ (directo) - 1

DEC @Ri

Byte: 1

Ciclos: 1

Operación: DEC

((Ri)) ⇐ ((Ri)) - 1

- **DIV AB**

Función: Divide.

Descripción: DIV AB divide los ocho bits enteros no signados en el Acumulador por los ocho bits no signados en el registro B El Acumulador recibe la parte entera del cociente; el registro B recibe el residuo de entero. La bandera de carry y OV se borran.

Excepción: Si B contenía originalmente 00H, los valores retornados en el Acumulador y en el registro B pueden ser indefinidos y la bandera de OV son setiadas. La bandera de carry se borra de todos modos

Ejemplo: El Acumulador contiene 251 (0FBH o 11111011B) y B contiene 18 (12H o 00010010B). La instrucción **DIV AB** retorna 13 en el Acumulador (0DH o 00001101B) y el valor 17 (11H o 00010001B) en B, ya que 251 = (13 * 18) + 17. Carry y OV son borrados.

Bytes: 1

Ciclos 4

Operación: DIV

(A)15-8 ⇐ (A)/(B)

(B)7-0 ↵

- **DJNZ <byte>, <rel-addr>**

Función: Decrementa y salta si no es cero.

Descripción: DJNZ decrementa en 1 la ubicación indicada en el primer operando, y salta a la ubicación indicada por el segundo operando si el valor resultante no es cero. Un valor original de 00H decrementará a 0FFH. Ninguna de las banderas será afectada. El destino del salto se computará agregando el valor del desplazamiento relativo signado en el último byte de instrucción del PC después de incrementar el PC al primer byte de la instrucción siguiente. La posición decrementada puede ser un

registro o byte directamente direccionado.

NOTA: Cuando ésta instrucción se usa para modificar la salida de un Puerto el valor usado como dato original del Puerto se lee desde el latch de datos de salida, no de los pines de entrada.

Ejemplo: Las posiciones 40, 50 y 60 de la RAM interna contienen los valores 01H, 70H y 15H respectivamente. La secuencia de instrucción,

```
DJNZ  40H,LABEL_1
DJNZ  50H,LABEL_2
DJNZ  60H,LABEL_3
```

producen un salto a la instrucción en el LABEL_2 con los valores 00H, 6FH y 15H en las tres posiciones de la RAM El primer salto no se produce porque el resultado es 0. Esta instrucción provee una manera simple de ejecutar un loop de programa un número determinado de veces, o para agregar un retardo moderado de tiempo (desde 2 a 512 ciclos de máquina) con una instrucción única. La secuencia de instrucción,

```
MOV       R2,#8
TOGGLE:   CPL       P1 7
DJNZ      R2,TOGGLE
```

ocho veces retornará a la instrucción TOGGLE P1.7, ocasionando cuatro pulsos de salida que aparecen en el bit 7 de salida del Puerto 1. Cada pulso dura tres ciclos de

máquina; dos para DJNZ y uno para alterar el pin.

DJNZ Rn,rel

Bytes: 2

Ciclos. 2

Operación: DJNZ

(PC) $\Leftarrow$ (PC) + 2

(Rn) $\Leftarrow$ (Rn) - 1

Si (Rn) > 0 o (Rn) < 0

entonces (PC) $\Leftarrow$ (PC) + rel

DJNZ directo,rel

Bytes: 3

Ciclos: 2

Operación: DJNZ

(PC) $\Leftarrow$ (PC) + 2

(directo) $\Leftarrow$ (directo) - 1

Si (directo) > 0 o (directo) < 0

entonces (PC) $\Leftarrow$ (PC) + rel

- **INC <byte>**

Función: Incrementa.

Descripción: INC incrementa la variable indicada en 1. Un valor original de 0FFH llevará a 00H. Ninguna de las banderas se afecta. Tres modos de direccionamiento se permiten: registro, directo, registro-indirecto.

NOTA: Cuando esta instrucción se usa para modificar una salida de Puerto, el valor usado como dato original del Puerto se lee desde el latch de datos de salida, no de los pines de entrada.

Ejemplo: El registro 0 contiene 7EH (01111110B). Las posiciones 7EH y 7FH de la RAM interna contienen 0FFH y 40H, respectivamente. La secuencia de instrucción,

INC @R0

INC R0

INC @R0

retornará el registro 0 con 7FH y las posiciones 7EH y 7FH retornarán respectivamente 00H y 41H

INC A

Byte: 1

Ciclos: 1

Operación: INC

(A) $\Leftarrow$ (A) + 1

INC Rn

Byte: 1

Ciclos. 1

Operación: INC

(Rn) $\Leftarrow$ (Rn) + 1

INC directo

Byte: 2

Ciclos: 1

Operación: INC

(directo) $\Leftarrow$ (directo) + 1

INC @Ri

Byte: 1

Ciclos. 1

Operación: INC

((Ri)) ⇐ ((Ri)) + 1

- **INC DPTR**

Función: Incrementa el Data Pointer.

Descripción: Incrementa los 16 bits del Data Pointer (Indicador de datos) en 1. Un incremento de 16 bit (módulo 2^{16}) es realizado; un sobreflujo (OV) del bit de menor orden del indicador de datos (DPL) de 0FFH hasta 00H, incrementando el byte de mayor orden (DPH) . Ninguna bandera se afecta. Este es el único registro de 16 bit que puede ser incrementado.

Ejemplo: Los registros DPH y DPL contienen 12H y 0FEH, respectivamente. La secuencia de instrucción,

INC DPTR

INC DPTR

INC DPTR

carga DPH y DPL con 13H y 01H

Bytes: 1

Ciclos: 2

Operación: INC

(DPTR) ⇐ (DPTR) + 1

- **JB bit,rel**

Función: Salta si el bit es uno.

Descripción: Si el bit es uno, salta a la dirección indicada; de otra manera continúe con la siguiente instrucción. El destino del salto se computa agregando el desplazamiento relativo signado en el tercer byte de instrucción de el PC, después de incrementar el PC al primer byte de la próxima instrucción. El bit examinado no es modificado. Ninguna de las banderas es afectada.

Ejemplo: El dato presente en la entrada del Puerto 1 es 11001010B. El Acumulador contiene 56 (01010110B) La secuencia de instrucciones,

JB P1.2,LABEL 1

JB ACC.2,LABEL 2

la ejecución del programa causa un salto a la instrucción identificada como LABEL 2.

Bytes: 3

Ciclos. 2

Operación: JB

(PC) ⇐ (PC) + 3

Si (bit) = 1

Entonces

(PC) ⇐ (PC) + rel

- **JBC bit,rel**

Función: Salta si el Bit es uno y resetea el bit.

Descripción: Si el bit indicado es uno, salta a la dirección indicada; de otra manera continúa con la próxima instrucción. El bit no se resetea si es un cero. El destino del salto se computa agregando el desplazamiento relativo signado en el tercer byte de instrucción de el PC, después de incrementar el PC al primer byte de la próxima instrucción. Ninguna de las banderas es afectada.

NOTA: Cuando esta instrucción es usada para probar un pin de salida, el valor usado como dato original se lee desde el latch de datos de salida, no del pin de entrada.

Ejemplo: El Acumulador contiene 56H (01010110B).

La secuencia de instrucción,

JBC ACC.3,LABEL 1

JBC ACC.2,LABEL 2

La ejecución del programa causa que continúe en la instrucción identificada por la etiqueta LABEL 2, con el Acumulador modificado en 52H (01010010B).

Bytes: 3

Ciclos: 2

Operación: JBC

(PC) $\Leftarrow$ (PC) + 3

Si (bit) = 1

entonces

(bit) $\Leftarrow$ 0

(PC) $\Leftarrow$ (PC) + rel

- **JC rel**

Función: Salta si el carry es uno.

Descripción: Si la bandera de carry es uno, salta a la dirección indicada; de otra manera continúa con la siguiente instrucción El destino del salto se computa agregando el desplazamiento relativo signado en el segundo byte de instrucción de el

PC, después se incrementa el PC dos veces. Ninguna de las banderas es afectada.

Ejemplo: La bandera de carry es resetiada.

La secuencia de instrucción,

JC LABEL 1

CPL C

JC LABEL 2

El carry es setiado y produce que la ejecución del programa continue en la instrucción identificada como LABEL 2

Bytes: 2

Ciclos: 2

Operación. JC

(PC) $\Leftarrow$ (PC) + 2

Si (C) = 1

entonces

(PC) $\Leftarrow$ (PC) + rel

- **JMP @A+DPTR**

Función: Salto indirecto.

Descripción: Adiciona el contenido de los 8 bit no signados de el Acumulador con los 16 bit del indicador de datos (data pointer), y carga la suma resultante al contador de programa. Esta sera la dirección para lograr la instrucción subsiguiente. Realiza la adición de 16 bit (módulo 2^{16}): Saca un carry de los ocho bits de menor orden propagándose hacia los bits de mayor orden. Ni el Acumulador ni el Indicador de Datos se alteran. Ninguna de las banderas es afectada.

Ejemplo: Un número par desde 0 a 6 esta en el Acumulador. La secuencia siguiente de instrucciones produce un salto a una de cuatro instrucciones AJMP en una tabla de saltos que comienza en JMP_TBL:

```
MOV     DPTR,#JMP_TBL
JMP     @A+DPTR
JMP_TBL     AJMP    LABEL0
AJMP    LABEL1
AJMP    LABEL2
AJMP    LABEL3
```

Si el Acumulador igula a 04H cuando comienza esta secuencia, ejecutará el salto hasta la etiqueta LABEL2. Recuerde que AJMP es una instrucción de dos bytes, así las instrucciones de salto comienzan todas en otras instrucciones.

Bytes: 1

Ciclos: 2

Operación: JMP

(PC) $\Leftarrow$ (A) + (DPTR)

- **JNB bit,rel**

Función: Salte si el bit es cero.

Descripción: Si el bit indicado es un cero, salta a la dirección indicada; de otra manera continúa con la siguiente instrucción. El destino del salto se computa agregando el desplazamiento relativo signado en el tercer byte de instrucción de el PC, después de incrementar el PC al primer byte de la próxima instrucción El bit indicado no es modificado. Ninguna de las banderas es afectada.

Ejemplo: El dato presente en la entrada del Puerto 1 es 11001010B. El Acumulador contiene 56 (01010110B) La secuencia de instrucción,

JNB P1.3,LABEL 1

JNB ACC.3,LABEL 2

causa que la ejecución del programa continúe en la instrucción identificada como LABEL2.

Bytes: 3

Ciclos: 2

Operación: JNB

(PC) $\Leftarrow$ (PC) + 3

Si (bit) = 0

entonces

(PC) $\Leftarrow$ (PC) + rel

- **JNC rel**

Función: Salta el carry es cero.

Descripción: Si la bandera de carry es un cero, salta a la dirección indicada; de otra manera continúa con la siguiente instrucción. El destino del salto es computado agregando el desplazamiento relativo signado en el segundo byte de instrucción de el PC, después se incrementa el PC dos veces para la próxima instrucción. La bandera

de carry no se modifica.

Ejemplo: La bandera de carry es setiada. La secuencia de instrucciones,

JNC LABEL 1

CPL C

JNC LABEL 2

resetea el carry y causa que la ejecución del programa continúe hasta la instrucción identificada con la etiqueta LABEL2.

Bytes: 2

Ciclos: 2

Operación: JNC

(PC) $\Leftarrow$ (PC) + 2

Si (C) = 0

entonces

(PC) $\Leftarrow$ (PC) + rel

- **JNZ rel**

Función: Salta si el Acumulador no es cero.

Descripción: Si cualquier bit del Acumulador es un 1, salta hasta la dirección indicada; de otra forma continúa con la próxima instrucción. El destino del salto es computado por la adición del desplazamiento relativo signado en el segundo byte de instrucción de el PC, después de incrementar el PC dos veces. Ninguna de las banderas es afectada.

Ejemplo: El Acumulador originalmente contiene 00H. La secuencia de instrucciones,

JNZ LABEL1

INC A

JNZ LABEL2

coloca en el Acumulador 01H y continúa en la etiqueta LABEL2.

Bytes: 2

Ciclos: 2

Operación: JNZ

(PC) $\Leftarrow$ (PC) + 2

Si (A) < > 0

entonces

(PC) $\Leftarrow$ (PC) + rel

- **JZ rel**

Función: Salta si el Acumulador es cero.

Descripción: Si todos los bits del Acumulador son cero, salta hasta la dirección indicada; de otra forma continúa con la próxima instrucción. El destino del salto es computado por la adición del desplazamiento relativo signado en el segundo byte de instrucción del PC, después de incrementar el PC dos veces. El Acumulador no se modifica. Ninguna de las banderas es afectada.

Ejemplo: El Acumulador originalmente contiene 01H. La secuencia de instrucciones,

```
JZ      LABEL1
DEC     A
JZ      LABEL2
```

cambia el Acumulador a 00H y causa que la ejecución del programa continúe en la instrucción identificada con la etiqueta LABEL2.

Bytes: 2

Ciclos: 2

Operación: JZ

(PC) $\Leftarrow$ (PC) + 2

Si (A) = 0

entonces

(PC) $\Leftarrow$ (PC) + rel

- **LCALL addr16**

Función: Llamado largo.

Descripción: LCALL llama una subrutina ubicada en la dirección indicada. La instrucción agrega tres al contador de programa para generar la dirección de la próxima instrucción y meter los 16 bits resultantes en la pila (primer byte bajo), incrementando el indicador de pila (stack pointer) por dos. Los bytes de mayor y menor orden del PC entonces se cargan respectivamente con el segundo y tercer byte de la instrucción LCALL. La ejecución del programa continúa con la instrucción en estas direcciones. La subrutina puede comenzar por lo tanto donde quiera dentro de los 64K bytes de espacio de dirección de memoria de programa. Ninguna de las banderas será afectada.

Ejemplo: Inicialmente el Stack Pointer será igual a 07H. La etiqueta "SUBRTN" se

asigna para localizar la memoria de programa en 1234H Después de ejecutar la instrucción **LCALL SUBRTN,** el Stack Pointer contendrá 09H en la ubicación 0123H, y las posiciones 08H y 09H de la RAM interna se cargaran con 26H y 01H, y el PC contendrá 1234H.

Bytes: 3

Ciclos: 2

Operación: LCALL

(PC) $\Leftarrow$ (PC) + 3

(SP) $\Leftarrow$ (SP) + 1

((SP)) $\Leftarrow$ (PC7-0)

(SP) $\Leftarrow$ (SP) + 1

((SP)) $\Leftarrow$ (PC15-8)

(PC) $\Leftarrow$ addr15-0

- **LJMP addr16**

Función: Salto largo.

Descripción: LJMP ocasiona un salto incondicional a la dirección indicada, cargando los bytes de mayor y menor orden del PC (respectivamente) con el segundo y tercer byte de instrucción. El destino puede por lo tanto ser donde quiera dentro de los 64K

bytes de espacio de dirección de memoria de programa. Ninguna de las banderas serán afectadas.

Ejemplo: La etiqueta "JMPADR" es asignada a la instrucción en la ubicación 1234H de la memoria de programa. La instrucción **LJMP JMPADR,** el contador de programa se carga con 1234H en la posición 0123H.

Bytes: 3

Ciclos: 2

Operación: LJMP

(PC) $\Leftarrow$ addr15-0

- **MOV <dest-byte>,<src-byte>**

Función: Mueva la variable byte.

Descripción: La variable byte indicada por el segundo operando es copiada en la posición especificada por el primer operando. El byte fuente no es afectado. Ninguno otro registro o bandera es afectada. Esta es en alto grado la operación más flexible. Quince combinaciones de modos de direccionamiento de fuente y destino se permiten.

Ejemplo: La posición 30H de la RAM interna contiene 40H. El valor de la posición 40H de la RAM es 10H. El dato presente en la entrada del Puerto 1 es 11001010B (0CAH).

```
MOV   R0,#30H     ;R0 <= 30H
MOV   A,@ R0      ;A <= 40H
MOV   R1,A        ;R1 <= 40H
MOV   B,@ R1      ;B <= 10H
MOV   @ R1,P1     ;RAM (40H) <= 0CAH
MOV   P2,P1       ;P2 # 0CAH
```

deja el valor 30H en el registro 0, 40H en el Acumulador y el registro 1 (en ambos), 10H en el registro B, y 0CAH (11001010B) en la posición 40H de la RAM y en la salida del Puerto 2.

MOV A,Rn

Bytes: 1

Ciclos. 1

Operación: MOV

(A) $\Leftarrow$ (Rn)

MOV A,directo

Bytes: 2

Ciclos: 1

Operación: MOV

(A) ⇐ (directo)

* MOV A,ACC no es una instrucción valedera.

MOV A,@Ri

Bytes 1

Ciclos: 1

Operación: MOV

(A) ⇐ ((Ri))

MOV A,#dato

Bytes: 2

Ciclos: 1

Operación: MOV

(A) ⇐ #dato

MOV Rn,A

Bytes: 1

Ciclos. 1

Operación: MOV

(Rn) ⇐ (A)

MOV Rn,directo

Bytes: 2

Ciclos. 2

Operación: MOV

(Rn) ⇐ (directo)

MOV @Ri,directo

Bytes: 2

Ciclos: 2

Operación: MOV

((Ri)) ⇐ (directo)

MOV Rn,#dato

Bytes: 2

Ciclos 1

Operación: MOV

(Rn) ⇐ #dato

MOV directo,Rn

Bytes: 2

Ciclos: 2

Operación: MOV

(directo) ⇐ (Rn)

MOV directo,@Ri

Bytes. 2

Ciclos: 2

Operación: MOV

(directo) ⇐ ((Ri))

MOV @Ri,A

Bytes: 1

Ciclos: 1

Operación: MOV

((Ri)) ⇐ (A)

MOV @Ri,#dato

Bytes: 2

Ciclos. 1

Operación MOV

((Ri)) ⇐ #dato

MOV directo,A

Bytes: 2

Ciclos: 1

Operación: MOV

(directo) ⇐ (A)

MOV directo,directo

Bytes: 3

Ciclos 2

Operación. MOV

(directo) ⇐ (directo)

MOV directo,#dato

Bytes: 3

Ciclos: 2

Operación: MOV

(directo) ⇐ #dato

- **MOV <dest-bit>,<src-bit>**

Función: Mueve el dato del bit.

Descripción: La variable booleana indicada por el segundo operando es copiada en la ubicación especificada por el primer operando. Uno de los operandos debe ser la bandera de carry; el otro puede ser cualquiera de los bit directamente direccionable. Ninguno de los otros registros o bandera es afectado.

Ejemplo: La bandera de carry es setiada originalmente. El dato presente en la entrada del Puerto 3 es 11000101B. El dato anteriormente escrito en la salida del Puerto 1 es 35H (00110101B). Las instrucciones,

MOV P1.3,C

MOV C,P3 3

MOV P1.2,C

el Carry saldrá resetiado y el Puerto 1 cambia a 39H (00111001B).

MOV C,bit

Bytes: 2

Ciclos: 1

Operación: MOV

(C) ⇐ (bit)

MOV bit,C

Byte: 2

Ciclos: 2

Operación: MOV

(bit) ⇐ (C)

- **MOV DPTR,#dato16**

Función: Carga el Data Pointer (Indicador de Datos) con una constante de 16 bit

Descripción: El Indicador de Datos (Data Pointer) es cargado con la constante de 16 bit indicada. El segundo byte (DPH) es el byte de mayor orden, mientras el tercer byte (DPL) contiene el byte de menor orden. Ninguna de las banderas es afectada. Esta es la instrucción única que mueve 16 bits de dato a la vez.

Ejemplo: La instrucción **MOV DPTR,#1234H,** carga el valor 1234H en el Indicador de Datos: DPH retendrá 12H y DPL retendrá 34H.

Bytes: 3

Ciclos: 2

Operación: MOV

(DPTR) ⇐ #dato15-0

DPH □ DPL ⇐ #dato15-8 □ #dato7-0

- **MOVC A,@a + <base-reg>**

Función: Mueve el código del byte.

Descripción: Las instrucciones MOVC cargan el Acumulador con un código de byte, o constante desde la memoria de programa. La dirección del byte traido es la suma de los 8 bit no signados originales contenidos en el Acumulador y el contenido de un registro base de 16 bit, que puede ser o el Indicador de Datos o el PC. En el segundo caso, el PC es incrementado a la dirección de la siguiente instrucción antes de ser adicionada con el Acumulador; de otra manera, el registro base no es alterado. La adición de 16 bit es hecha de modo que los 8 bits de menor orden sacan un carry que se propaga hacia los bits de mayor orden Ninguna de las banderas es afectada.

Ejemplo: Un valor entre 0 y 3 está en el Acumulador. La siguiente instrucción traducirá el valor en el Acumulador a uno de cuatro valores definido por la orden del

DB (define byte).

```
REL_PC:     INC     A
MOVC    A,@A+PC
RET
DB      66H
DB      77H
DB      88H
DB      99H
```

si la subrutina se llama con el Acumulador igual a 01H, retornará con 77H en el Acumulador. La instrucción INC A antes de la instrucción MOVC se necesita para saltar sobre la instrucción RET de la tabla anterior. Si los diferentes bytes de código separan el MOVC de la tabla, el número correspondiente se agregaría al Acumulador en lugar de ello.

MOVC A,@A+DPTR

Bytes: 1

Ciclos: 2

Operación: MOVC

(A) $\Leftarrow$ ((A) +(DPTR))

MOVC A,@A+PC

Bytes: 1

Ciclos: 2

Operación: MOVC

(PC) ⇐ (PC) + 1

(A) ⇐ ((A) +(PC))

- **MOVX <dest-byte>,<src-byte>**

Función: Movimiento externo.

Descripción: Las instrucciones MOVX transfieren datos entre el Acumulador y un byte de datos de memoria externa, de aquí en adelante la "X" se añade a MOV. Hay dos tipos de instrucciones distinguiendose entre ellas por proveer 8 o 16 bits de dirección indirecta a los datos de RAM externa.

En el primer tipo, los contenidos de R0 o R1 en el banco actual de registro provee una dirección de 8 bit multiplexada con datos sobre P0. Ocho bits son suficientes para la expansión decodificada de I/O externa o para un arreglo de RAM relativamente pequeño. Para arreglos algo más grandes, cualquiera de los pines de salida del Puerto puede ser usado como salida de mayor orden de los bits de dirección. Estos pines

serían controlados por una instrucción de salida que precede a MOVX.

En el segundo tipo de instrucción MOVX, el indicador de dato genera una dirección de 16 bit. P2 entrega los ocho bits de dirección de mayor orden (los contenidos de DPH) mientras que P0 multiplexa los bits de menor orden (DPL) con datos. El Registro P2 de Funciones Especiales contiene su contenido previo mientras los buffers de salida de P2 emiten los contenido de DPH. Esta forma es más rápida y eficiente cuando entran arreglos muy grandes de datos (de hasta 64K bytes), ninguna instrucción adicional se necesita para establecer la salida de los Puertos.

Es posible en algunas situaciones mezclar los tipos de MOVX. Un arreglo grande con manejo de linea de dirección de mayor orden a P2 seguido por la instrucción MOVX usando R0 o R1.

Ejemplo Una RAM externa de 256 bytes que usa lineas multiplexadas de dirección/datos (ej., un Intel 8155 RAM/I/O/TIMER) se conecta al Puerto 0 del 8051. El Puerto 3 provee lineas de control para la RAM externa. Los Puertos 1 y 2 son usados como I/O normales. Los registros 0 y 1 contienen 12H y 34H. La posición 34H de la RAM externa contiene el valor 56H. La secuencia de instrucción,

```
MOVX    A,@R1
MOVX    @R0,A
```

copia el valor 56 en ambos. El Acumulador y la posición 12H de la RAM externa.

MOVX A,@Ri

Bytes: 1

Ciclos. 2

Operación: MOVX

(A) ⇐ ((Ri))

MOVX A,@DPTR

Bytes: 1

Ciclos: 2

Operación: MOVX

(A) ⇐ ((DPTR))

MOVX @Ri,A

Bytes: 1

Ciclos. 2

Operación. MOVX

((Ri)) ⇐ (A)

MOVX @DPTR,A

Bytes: 1

Ciclos· 2

Operación. MOVX

((DPTR)) $\Leftarrow$ (A)

- **NOP**

Función: Ninguna Operación.

Descripción: La ejecución continúa en la instrucción siguiente. A excepción del PC, ninguna de los registros o banderas es afectado.

Ejemplo: Se desea que produzca un pulso de salida bajo en el bit 7 del Puerto 2 duradero exactamente 5 ciclos Una secuencia SETB/CLR simple generaría un pulso durante un ciclo, sólo cuatro ciclos adicionales deben meterse. Esto puede hacerse (asumamos que ninguna interrupción es permitida) con la secuencia de instrucción,

```
CLR     P2.7
NOP
NOP
NOP
NOP
SETB    P2.7
```

Bytes: 1

Ciclos. 1

Operación: NOP

(PC) $\Leftarrow$ (PC)+1

- **MUL AB**

Función: Multiplica.

Descripción: MUL AB multiplaca los ocho bits enteros no signados en el Acumulador por los ocho bits no signados en el registro B. El byte de menor orden del producto de 16 bits se deja en el Acumulador, y el byte de mayor orden en el registro B. Si el producto es mayor de 255 (0FFH) la bandera de desborde (OV) es setiada; de otra manera es resetiada. La bandera de carry es siempre resetiada.

Ejemplo: Originalmente el Acumulador contiene el valor 80 (50H). El registro B contiene el valor 160 (0A0H). La instrucción **MUL AB** el producto da 12800 (3200H), por lo tanto, B cambia a 32H (00110010B) y el Acumulador se resetea. La bandera de OV es setiada y el Carry es resetiado.

Bytes: 1

Ciclos: 4

Operación: MUL

(A)7-0 ⇐ (A)*(B)

(B)15-8 ⌟

- **ORL <dest-byte>,<src-byte>**

Función: Operación Lógica OR para bit variable.

Descripción: ORL desempeña la operación discreta OR lógica entre las variables indicadas, almacenando los resultados en el byte de destino. Ninguna de las banderas es afectada.

Los dos operandos permiten seis combinaciones de modo de direccionamiento. Cuando el destino es el Acumulador, la fuente puede usar direccionamiento de: registro, directo, registro-indirecto, o inmediato; cuando el destino es una dirección directa la fuente puede ser el Acumulador o un dato inmediato.

NOTA: Cuando esta instrucción es usada para modificar la salida del Puerto, el valor usado como dato original del Puerto se leerá desde el latch de datos de salida, no de los pines de entrada.

Ejemplo: Si el Acumulador contiene 0C3H (11000011B) y R0 contiene 55H

(01010101B) entonces la instrucción **ORL A,R0** retorna el Acumulador cargado con el valor 0D7H (11010111B). Cuando el destino es un byte directamente direccionado, la instrucción puede colocar combinaciones de bits en cualquier posición de RAM o registro del0 hardware. Los bits setiados están determinados por un byte de mascara, que puede ser un dato de valor constante en la instrucción o una variable computada en el Acumulador en el tiempo de ejecución. La instrucción **ORL P1,00110010B** colocando los bits 5, 4 y 1 de salida del Puerto 1 en uno.

ORL A,Rn

Bytes: 1

Ciclos: 1

Operación: ORL

(A) ⇐ (A) V (Rn)

ORL A,@Ri

Bytes: 1

Ciclos: 1

Operación. ORL

(A) ⇐ (A) V ((Ri))

ORL A,directo

Bytes: 2

Ciclos: 1

Operación. ORL

(A) ⇐ (A) V (directo)

ORL A,#dato

Bytes: 2

Ciclos: 1

Operación. ORL

(A) ⇐ (A) V #dato

ORL directo,#dato

Bytes: 3

Ciclos: 2

Operación· ORL

(directo) ⇐ (directo) V #dato

ORL directo,A

Bytes: 2

Ciclos: 1

Operación: ORL

(directo) ⇐ (directo) V (A)

- **ORL C,<src-bit>**

Función: Función lógica OR para la variable bit.

Descripción: Setea la bandera de carry si el valor booleano es un 1 lógico; de otro modo deja la bandera de carry en su estado actual. Un slash ("/") precediendo el operando en el Lenguaje Assembly indica que el complemento lógico de el bit direccionado es usado como valor de fuente, pero el bit fuente en sí mismo no es afectado. Ninguna otra bandera es afectada.

Ejemplo: La bandera de carry es setiada si y solo si P1.0 = 1, ACC.7 = 1 o OV = 0:

```
MOV   C,P1.0      ;carga el Carry con el pin de entrada P10
ORL   C,ACC.7     ;o Carry con el ACC.BIT7
ORL   C,/OV       ;o Carry con el inverso de OV.
```

ORL C,bit

Bytes: 2

Ciclos: 2

Operación. ORL

(C) $\Leftarrow$ (C) V (bit)

ORL C,/bit

Bytes: 2

Ciclos: 2

Operación: ORL

(C) $\Leftarrow$ (C) V (bit)

- **POP directo**

Función: Sacar del stack.

Descripción: Los contenidos de la RAM interna direccionados por el Stack Pointer son leidos, y el Stack Pointer es decrementado en uno. El valor leido entonces se transfiere al byte directamente direccinado que se indica. Ninguna de las banderas es afectada

Ejemplo: El Stack Pointer originalmente contiene el valor 32H, y las posiciones 30H hasta la 32H contienen los valores 20H, 23H y 01H respectivamente. La secuencia de instrucciones,

POP DPH

POP DPL

retorna el Stack Pointer igual a 30H y el Data Pointer a 0123H. En este punto la

instrucción **POP SP,** retornará el Stack Pointer en 20H. Note que en este caso especial el Stack Pointer era decrementado a 2FH antes de ser cargado con el valor 20H.

Bytes: 2

Ciclos: 2

Operación: POP

(directo) ⇐ ((SP))

(SP) ⇐ (SP) - 1

- **PUSH directo**

Función: Meter en el Stack.

Descripción: El Stack Pointer es incrementado en 1. El contenido de la variable indicada es copiado en la posición de la RAM interna direccionado por el Stack Pointer. De otra manera, ninguna de las banderas es afectada.

Ejemplo: Al entrar en una rutina de interrupción el Stack Pointer contiene 09H. El Data Pointer contiene el valor 0123H. La secuencia de instrucciones,

PUSH DPL

PUSH DPH

retornará el Stack Pointer con 0BH y almacena 23H y 01H en las posiciones 0AH y 0BH de la RAM interna respectivamente.

Bytes: 2

Ciclos. 2

Operación: PUSH

(SP) $\Leftarrow$ (SP) + 1

((SP)) $\Leftarrow$ (directo)

- **RET**

Función: Retorna de subrutina.

Descripción. RET saca los bytes de mayor y menor orden del PC consecutivamente desde el stack decrementando el Stack Pointer en dos. La ejecución del programa continúa en la dirección resultante, generalmente la instrucción inmediatamente siguiente a un ACALL o LCALL. No afecta ninguna de las banderas.

Ejemplo: El Stack Pointer originalmente contiene 0BH Las posiciones 0AH y 0BH contienen los valores 23H y 01H respectivamente. La instrucción **RET,** entrega el

Stack Pointer igual a 09H y retorna la ejecución del programa en la posición 0123H.

Bytes: 1

Ciclos: 2

Operación. RET

(PC15-8) ⇐ ((SP))

(SP) ⇐ (SP) - 1

(PC7-0) ⇐ ((SP))

(SP) ⇐ (SP) - 1

- **RETI**

Función: Retorna de interrupción.

Descripción: RETI saca los bytes de mayor y menor orden del PC consecutivamente desde el stack, y restaura la interrucción lógica para aceptar interrucciones adicionales en el mismo nivel de prioridad en el momento justo de un proceso. El Stack Pointer es decrementado por dos. Ninguno de los otros registros es afectado; el PSW no es restaurado automáticamente a su condición de antes de la de interrupción. La ejecución del programa continúa en la dirección resultante, que es generalmente la instrucción inmediatamente siguiente al punto en que la interrupción pedida es detectada. Si una interrupción de más bajo nivel estaba pendiente cuando la

instrucción RETI se ejecutaba, esa instrucción será ejecutada antes de que la interrupción pendiente sea procesada.

Ejemplo: El Stack Pointer originalmente contiene el valor 0BH. Una interrrupción se detecta durante la instrucción que termina en la ubicación 0122H. Las posiciones 0AH y 0BH de la RAM interna contienen los valores 23H y 01H respectivamente. La instrucción **RETI** retornará el Stack Pointer con un valor igual a 09H y la ejecución del programa continuará en la posición 0123H.

Bytes: 1

Ciclos. 2

Operación: RETI

(PC15-8) $\Leftarrow$ ((SP))

(SP) $\Leftarrow$ (SP) - 1

(PC7-0) $\Leftarrow$ ((SP))

(SP) $\Leftarrow$ (SP) - 1

- **RL A**

Función: Rota el Acumulador a la izquierda.

Descripción. Los ocho bits en el Acumulador son rotados un bit a la izquierda. El bit 7 se rota hacia la posición del bit 0. Ninguna de las banderas es afectada.

Ejemplo: El Acumulador contiene el valor 0C5H (11000101B). La instrucción **RL A,** retorna el Acumulador cargado con el valor 8BH (10001011B) sin afectar el Carry.

Bytes: 1

Ciclos. 1

Operación: RL

$(An + 1) \Leftarrow (An) \quad n = 0\text{-}6$

$(A0) \Leftarrow (A7)$

- **RLC A**

Función: Rota el Acumulador a la izquierda a través de la bandera de carry.

Descripción: Los ocho bits en el Acumulador y la bandera de carry son al mismo tiempo rotados un bit a la izquierda. El bit 7 se mueve hacia la bandera de carry; el valor original de la bandera de carry se mueve hacia la posición del bit 0. Ninguna de

las otras banderas es afectada.

Ejemplo: El Acumulador contiene el valor 0C5H (11000101B), y el carry es cero. La instrucción **RLC A,** deja el Acumulador con el valor 8AH (10001010B) con el carry puesto en 1.

Bytes: 1

Ciclos. 1

Operación: RLC

(An + 1) $\Leftarrow$ (An) n = 0-6

(A0) $\Leftarrow$ (C)

(C) $\Leftarrow$ (A7)

- **RR A**

Función: Rota el Acumulador a la derecha.

Descripción: Los ocho bits en el Acumulador son rotados un bit a la derecha. El bit 0 se rota hacia la posición del bit 7. Ninguna de las banderas es afectada.

Ejemplo: El Acumulador contiene el valor 0C5H (11000101B). La instrucción **RR A** deja el Acumulador con el valor 0E2H (11100010B) sin afectar el carry.

Bytes: 1

Ciclos: 1

Operación: RR

(An) $\Leftarrow$ (An + 1) n = 0-6

(A7) $\Leftarrow$ (A0)

- **RRC A**

Función: Rota el Acumulador a la derecha a través de la bandera de Carry.

Descripción: Los ocho bits en el Acumulador y la bandera de carry son al mismo tiempo rotados un bit a la derecha. El bit 0 se mueve hacia la bandera de carry; el valor original de la bandera de carry se mueve hacia la posición del bit 7. Ninguna de las banderas es afectada.

Ejemplo. El Acumulador contiene el valor 0C5H (11000101B), y el carry es cero. La instrucción **RRC A** deja el Acumulador con el valor 62 (01100010B) con el carry puesto en 1.

Bytes: 1

Ciclos. 1

Operación: RRC

(An) ⇐ (An + 1) n = 0-6

(A7) ⇐ (C)

(C) ⇐ (A0)

•

- **SETB <bit>**

Función: Setea Bit.

Descripción. SETB fija el bit indicado en 1. SETB puede operarar sobre la bandera de carry o directamente en cualquier bit direccionable. Ninguna otra bandera es afectada.

Ejemplo: La bandera de carry es resetiada. La salida del Puerto 1 se ha escrito con el valor 34H (00100100B). Las instrucciones,

SETB C

SETB P1.0

deja la bandera de carry puesta en 1 y el dato cargado en la salida del Puerto 1 es 35H (00110101B).

SETB C

Bytes: 1

Ciclos. 1

Operación SETB

(C) $\Leftarrow$ 1

SETB bit

Bytes: 2

Ciclos. 1

Operación: SETB

(bit) $\Leftarrow$ 1

- **SJMP rel**

Función: Salto corto.

Descripción: El control de programa salta incondicionalmente a la dirección indicada. El destino del salto se computa agregando el desplazamiento signado en el segundo byte del PC, después se incrementa el PC dos veces. Por lo tanto, el rango de destino permitido es de 128 bytes.

Ejemplo: La etiqueta "RELADR" se asigna a una instrucción en la posición de

memoria de programa 0123H. La instrucción **SJMP RELADR,** salta y se posiciona dentro de la dirección 0100H. Después la instrucción es ejecutada y el PC contendrá el valor 0123H.

(NOTA: Bajo condiciones extremas la instrucción siguiente a SJMP estará en 102H. Por lo tanto, el byte desplazado de la instrucción será la compensación relativa (0123H-0102H)=21H. De otra manera, un SJMP con un desplazamiento de OFEH podria ser un loop infinito de una instrucción).

Bytes: 2

Ciclos: 2

Operación: SJMP

(PC) $\Leftarrow$ (PC) + 2

(PC) $\Leftarrow$ (PC) + rel

- **SUBB A,<src-byte>**

Función: Reste pidiendo prestado.

Descripción: SUBB resta la variable indicada y la bandera de carry juntamente con el Acumulador y deja el resultado en el Acumulador. SUBB setea la bandera de carry si

el bit 7 necesita pedir prestado, y resetea C de otra manera. (Si C está puesta en uno antes de ejecutar una instrucción SUBB, esto indica que fue necesario pedir prestado para la instrucción anterior en una resta múltiple de precisión, de modo que el carry se resta desde el Acumulador conjuntamente con el operando fuente). AC es puesta en uno si el bit 3 necesita pedir prestado, de otra manera es resetiada. OV es puesta en uno si el bit 6 necesita pedir prestado pero no el bit 7, o en el bit 7, pero no en el bit 6. Cuando se restan enteros signados, OV indica que un número negativo es producido cuando un valor negativo es restado de un valor positivo, o un resultado positivo cuando un número positivo es restado de un número negativo.

El operando fuente permite cuatro modos de direccionamiento: registro, directo, registro-indirecto, o inmediato

Ejemplo: El Acumulador contiene 0C9H (11001001B), el registro contiene 54H (01010100B), y la bandera de carry es puesta en uno. La instrucción **SUBB A,R2,** retornará 74H (01110100B) en el Acumulador con las banderas de Carry y AC resetiadas pero OV puesta en uno.

Nótese que 0C9H menos 54H es igual a 75H. La diferencia entre este resultado y el anterior, se debe a que la bandera de carry es setiada antes de comenzar la operación. Si el estado del carry es desconocido antes de comenzar una resta simple o múltiple de

precisión, explicitamente se debería resetiar con una instrucción CLR C.

SUBB A,Rn

Bytes: 1

Ciclos: 1

Operación: SUBB

(A) ⇐ (A) - (C) - (Rn)

SUBB A,directo

Bytes: 2

Ciclos: 1

Operación: SUBB

(A) ⇐ (A) - (C) - (directo)

SUBB A,@Ri

Bytes: 1

Ciclos: 1

Operación: SUBB

(A) ⇐ (A) - (C) - ((Ri))

SUBB A,#dato

Bytes: 2

Ciclos: 1

Operación: SUBB

(A) ⇐ (A) - (C) - #dato

- **SWAP A**

Función: Intercambia nibbles dentro del Acumulador.

Descripción: SWAP A intercambia los nibbles de menor y mayor orden (cuatro bits por nibble) del Acumulador (bits 3-0 y bits 7-4). La operación puede también ser como si se rotara cuatro bits de la instrucción. Ninguna de las banderas es afectada.

Ejemplo: El Acumulador contiene el valor 0C5H (11000101B). La instrucción **SWAP A,** retornará el Acumulador cargado con el valor 5CH (01011100B).

Bytes: 1

Ciclos: 1

Operación: SWAP

(A3-0) ⇔ (A7-4)

- **XCH A<byte>**

Función: Intercambia el Acumulador con la variable byte.

Descripción. XCH carga el Acumulador con los contenidos de la variable indicada, y simultaneamente escribe los contenidos originales del Acumulador a la variable indicada. Los operandos fuente/destino pueden usar direccionamiento de: registro, directo, o registro indirecto.

Ejemplo: R0 contiene la dirección 20H. El Acumulador contiene el valor 3FH (00111111B). La posición 20H de la RAM interna contiene el valor 75H (01110101B). La instrucción **XCH A,@R0,** retorna la posición 20H de la RAM interna con 3FH (00111111B) y 75H (01110101B) en el Acumulador.

XCH A,Rn

Bytes: 1

Ciclos: 1

Operación: XCH

(A) $\Leftrightarrow$ (Rn)

XCH A,directo

Bytes: 2

Ciclos. 1

Operación: XCH

(A) ⇔ (directo)

XCH A,@Ri

Bytes: 1

Ciclos· 1

Operación: XCH

(A) ⇔ ((Ri))

- **XCHD A,@Ri**

Función: Intercambia dígito.

Descripción. XCHD cambia el nibble de menor orden del Acumulador (bits 3-0, generalmente representando un digito hexadecimal o BCD) con la posición de la RAM interna direccionada indirectamente por el registro especificado. Los nibbles de mayor orden (bit 7-4) de cada registro no son afectados. Ninguna de las banderas es afectada.

Ejemplo: R0 contiene la dirección 20H El Acumulador contiene el valor 36H (00110110B). La posición 20H de la RAM interna contiene el valor 75H (01110101B). La instrucción **XCHD A,@R0,** la posición 20H de la RAM interna retornará el valor 76H (01110110B) y 35H (00110101B) en el Acumulador.

Bytes: 1

Ciclos: 1

Operación: XCHD

(A3-0) ⇔ ((Ri3-0))

- **XRL <dest-byte>,<src-byte>**

Función: Operación lógica OR-Exclusiva para la variable byte.

Descripción: XRL desempeña la operación lógica OR-Exclusiva entre las variables indicadas, almacenando el resultado en el destino. Ninguna de las banderas es afectada.

Los dos operandos permiten seis combinaciones de modos de direccionamiento. Cuando el destino es el Acumulador, la fuente puede usar direccionamiento de: registro, directo, registro indirecto, o inmediato; cuando el destino es una dirección

directa la fuente puede ser el Acumulador o datos inmediatos.

NOTA: Cuando esta instrucción es usada para modificar una sallida de Puerto, el valor usado como dato original del Puerto se lee desde el latch de datos de salida, no de los pines de entrada.

Ejemplo: Si el Acumulador contiene 0C3H (11000011B) y el registro 0 contiene 0AAH (10101010B) entonces la instrucción **XRL A,R0** ,retornará el Acumulador con el valor 69H (01101001B). Cuando el destino es un byte directamente direccionado esta instrucción puede complementar combinaciones de bits en cualquier posición de la RAM o registro del hardware. El modelo de bits para ser complementado es determinado entonces por un byte de mascara, o una constante contenida en la instrucción o una variable computada en el Acumulador en el tiempo de ejecución. La instrucción **XRL P1,#00110001B,** complementa los bits 5, 4 y 0 de la salida del Puerto 1.

XRL A,directo

Bytes. 2

Ciclos. 1

Operación: XRL

(A) $\Leftarrow$ (A) $\forall$ (directo)

XRL A,Rn

Bytes 1

Ciclos. 1

Operación: XRL

(A) $\Leftarrow$ (A) $\forall$ (Rn)

XRL A,@Ri

Bytes: 1

Ciclos: 1

Operación: XRL

(A) $\Leftarrow$ (A) $\forall$ ((Ri))

XRL A,#dato

Bytes 2

Ciclos: 1

Operación: XRL

(A) $\Leftarrow$ (A) $\forall$ #dato

XRL directo,A

Bytes: 2

Ciclos: 2

Operación XRL

(directo) $\Leftarrow$ (directo) $\forall$ (A)

XRL directo,#dato

Bytes: 3

Ciclos: 2

Operación: XRL

(directo) $\Leftarrow$ (directo) $\forall$ #dato

4.0. MANUAL DE EXPERIMENTOS

4.1. PRACTICA No 1: Manejo y Conocimiento del Set de Instrucciones del Sistema de Desarrollo Microlab 51 y su Software de Comunicaciones

4.1.1. Objetivo General

Conocer el manejo del Sistema de Desarrollo Microlab 51

4.1.2. Objetivos Específicos

- Instruir al usuario en el manejo del menú y las principales funciones del Microlab 51.
- Mostrar la secuencia a seguir para enviar y recibir archivos usando el software de comunicaciones MICROWIN.EXE.
- Conocer las herramientas adicionales al software de comunicaciones.

- Analizar las ventajas que ofrece la utilización del computador para el diseño y la depuración de programas

4.1.3. Manejo de Las Herramientas Básicas Del Microlab 51

Realice la conexión del Microlab 51 con el adaptador de voltaje dispuesto para el equipo.

A continuación se presenta el menú principal del sistema de desarrollo Microlab 51, el cual aparece después de presionada una tecla, al momento de alimentar el equipo o cuando se produce un reset por parte del usuario.

Edit	Asmb	Debug
Com	Run	Ver

Cualquier opción de este menú se puede seleccionar presionando la letra mayúscula resaltada o simplemente ubicándose con las flechas de desplazamiento sobre la opción y oprimiendo la tecla Enter (↵).

La siguiente son las funciones de cada una de las opciones de este menú (menú principal):

Edit: Se utiliza para editar programas fuentes en lenguaje ensamblador.

Asmb: Se utiliza para convertir el programa fuente en lenguaje de máquina entendible por el microcontrolador (hexadecimal), este programa ha sido digitado directamente en el editor del Microlab 51 o enviado desde un computador hasta el equipo.

Debug: Se utiliza para la depuración de programas.

Com: Se utiliza para la comunicación del Microlab 51 con un computador.

Run: Se utiliza para la ejecución de programas en tiempo real.

4.1.3.1. Suma de dos Registros

Ubicado en el menú principal, presione la tecla **E** (Edit) y digite el siguiente programa.

```
ORG   9200H       ;DIRECCIÓN DE INICIO DEL PROGRAMA.
CLR   C           ;INICIALIZA EL CARRY EN CERO
MOV   R0,#0A1H    ;INICIALIZA EL REGISTRO R0
MOV   A,#0F6H     ;INICIALIZA EL REGISTRO ACC
ADDC  A,R0        ;SUMA EL REGISTRO A CON EL REGISTRO R0
LJMP  80H
```

Nota: Preferiblemente el inicio de los programas se debe direccionar a la posición 9200H, la cual está dispuesta para el usuario (refiérase al manual de usuario del Microlab 51).

Cuando termine de digitar el programa, presione la tecla escape (ESC) para salir del editor de texto.

Presione la tecla A (Asmb), esta opción revisa si hay errores de sintaxis y traduce a lenguaje de máquina cada uno de las líneas digitadas; si el programa tiene esta clase de errores, aparecerá en el despliege el mensaje **ERR Assembly ...**, y se debe seguir el siguiente procedimiento:

- Presione la tecla escape (ESC).
- Presione la tecla E (Edit) para ingresar al editor de texto.
- Presione las teclas control S (CTRL + S).

Después de realizado este proceso aparecerá en el despliege el listado de errores y la línea en la cual se presentó. Diríjase a la tabla #11.4 en el manual de usuario del Microlab 51 la cual contiene el significado de cada uno de los códigos de error.

Si no hay ningún error de sintaxis, mostrara el mensaje **End Assembly... OK** (el cual indica que la revisión ha sido satisfactoria).

Sugerencia:

Si el programa no ha tenido ningún problema de sintaxis, es preciso que experimente el siguiente paso, el cual le dará una clara idea de lo que debe hacer cuando estos se presentan.

- En la línea #4 borre de la instrucción MOV la letra V usando la tecla de retroceso o las teclas CTRL D, presione la tecla escape y compile nuevamente el programa
- Corrija los errores de la manera indicada anteriormente.

Por último presione la tecla R (Run), para ejecutar su programa.

Esta es la forma en la cual se debe proceder para digitar y ejecutar un programa en el Microlab 51.

Ejercicio propuesto: Borre la línea #6 **AJMP 80H**, y ejecute nuevamente su programa. Explique cuál es la diferencia entre cada programa, y que hace la línea **AJMP 80H.**

4.1.3.2 Operaciones Aritméticas del Microcontrolador

Para continuar con el proceso de reconocimiento y manejo del equipo digite el siguiente programa, el cual usa todas las funciones aritméticas del microcontrolador.

```
;****** OPERACIONES ARITMÉTICAS ******
;****** DECLARACIÓN DE VARIABLES ******

RESULTA     EQU   50H
RESULTB     EQU   51H
RESULTC     EQU   52H
RESULTD     EQU   53H
```

```
RESULTE     EQU   54H
SUMA        EQU   55H
REST        EQU   56H
MULT        EQU   57H
DIVI        EQU   58H

;**** PROGRAMA PRINCIPAL ******

;****** SUMA ******
            ORG         9200H
INI         CLR         C           ;Inicializa el carry en cero
            MOV         R0,#0A1H    ;Inicializa el registro R0
            MOV         A,#0A1H     ;Inicializa el registro ACC
            ADDC        A,R0        ;Suma el registro a y el registro R0
            MOV         RESULTA,ACC ;Guarda el ACC en la Posmem 50H
            MOV         SUMA,PSW    ;Guarda el Registro PSW en 51H
;****** RESTA ******
            MOV         A,#07H
            MOV         R1,#08H     ;Inicializa el Registro R1
            SUBB        A,R1
            MOV         RESULTB,ACC ;Guarda en la Posmem 52H
            MOV         REST,PSW    ;Guarda en la Posmem 53H
            ADDC        A,#05H
            MOV         RESULTC,ACC
            LJMP        80H
            END
```

Ejecute el programa y verifique los resultados.

Ejercicio Propuesto: Utilice la función de multiplicar (MULT) y de dividir registros (DIV) de la misma forma como se usó ADDC y SUBB, y evalúe su programa.

4.1.3.3. Depuración de Programas en el Sistema de Desarrollo Microlab 51

Uno de las principales funciones que tiene cualquier lenguaje de programación (en este caso el programa monitor del Microlab 51), es la facilidad que este pueda tener para la depuración de programas, lo cual permite evaluar el programa paso a paso y corregir posibles errores que este pudiera tener.

A continuación se mostrará la manera como se puede hacer depuración de los programas en el Sistema de Desarrollo Microlab 51.

En el menú principal, digite la tecla D (Debug); el siguiente menú aparecerá en el despliegue:

Unasm	Enter	Dump
Block	Regist	Trace

Con el menú anterior usted podrá realizar funciones como observar y/o modificar memoria y ejecutar el programa línea por línea. Aquí se presentan las principales características de estas funciones.

Unsam: Desensambla los programas código que se encuentran en memoria, es decir cambia el programa que está en lenguaje de máquina (hexadecimal) a lenguaje

ensamblador.

Enter: Permite visualizar y/o modificar el contenido de memoria interna y/o externa del Microlab 51.

Dump: Tiene la misma función que la opción **Enter**, con la diferencia de que sólo se puede visualizar el contenido de la memoria, es decir no permite la modificación de ésta.

Block: Se utiliza para funciones que involucran la memoria de datos (interna/externa) y de programa tales como llenar un bloque de memoria (Fill), mover un bloque (Move), imprimir (Print) y buscar cadenas (Search), estas opciones se encuentran agrupadas en un submenú.

Register: Esta opción permite visualizar el contenido de los registros básicos del Microcontrolador 80C31.

Trace: Opción utilizada para ejecutar el programa paso a paso, lo cual permite hacer un seguimiento al flujo del programa.

El siguiente proceso le indicará como depurar un programa, y como revisar las variables y/o registros que intervienen en el programa.

- Presione la tecla R (Regist), observe el contenido de los registros, compare si el valor del acumulador A (ACC) corresponde realmente al resultado esperado; con las flechas arriba-abajo (↑ ↓) podrá observar el contenido de los registros R0 al R7.

- Presione la tecla D (Dump) se desplegará un menú con el cual se le puede indicar al Microlab 51 la dirección y el tipo de memoria que se desea visualizar.

Prg	**D**at	**I**nt	Address
( )	(*)	( )	

- Digite la letra D (Datos), escriba la dirección 0050 y presione la tecla Enter (↵). Como puede observar, en el despliegue se muestra la dirección y el contenido de la memoria.

- Revise si el contenido de dichas posiciones de memoria corresponde al valor que debería tener el acumulador en cada operación, y al del PSW (Program Status Word), el cual indica el estado de las banderas del microcontrolador (carry, overflow, paridad, etc).

¿Por qué es importante conocer el valor del **PSW**.?

¿Cómo se podría utilizar la información del **PSW** en un proceso de control.?

Presione la tecla escape (ESC) para salir del menú Dump y regresar al menú anterior.

- Digite la tecla T (Trace) para correr el programa paso a paso. Antes de poder depurar el programa el Microlab 51 le preguntará por la dirección de memoria donde se encuentra localizado el inicio del programa (en este caso la posición es 9200H)

como se muestra en el siguiente cuadro:

New value PC > 9200

Después de presionar la tecla Enter (↵), aparecerá nuevamente el contenido de los registros en la pantalla; presione la tecla L y observe como se ejecuta el programa paso a paso (línea a línea) cada vez que se presiona dicha tecla.

Nota: En cuanto a las restricciones del programa Trace, refiérase a la página 71 apartado 11.15.3.6.1 del manual del usuario.

Ejercicio Propuesto: Use el programa del numeral 3.2 y evalúelo para diferentes valores de **R0, ACC**, etc., de tal manera que pueda observar las variaciones del **PSW**.

¿Cuándo se presenta desbordamiento y que pasa con este efecto?

4.1.3.4 Manejo del Microlab 51 para Comunicaciones con un Computador

Microlab 51 viene dispuesto con un puerto serial (RS232)[1] conector tipo DB9 que le permite realizar comunicaciones con un periferico. El siguiente procedimiento le permitirá conocer como comunicarse con un computador:

[1] Protocolo de comunicación RS232

- Digite el programa del apartado anterior.

- Conecte el Microlab 51 a un computador a través del puerto serial con el cable DB9 a DB25.

- Ingrese al programa para comunicaciones (**MICROWIN.EXE**). El computador debe tener instalado el "Sistema Operacional Microsoft Windows" versión 3.1 o superior. Para el manejo del Programa utilice la opción **Ayuda** en el menu principal o diríjase al Manual de Usuario.

- Escoja la opción **Com** desde el menú principal en el Microlab 51. El siguiente menú aparecerá en el despliegue:

Conf	Receiv	Xmit
rxBin	rxTxt	reM

En este menú se encuentran todas las opciones que trae el Microlab 51 para comunicaciones

Conf: Permite configurar el modo de transmisión y la velocidad de transmisión.

Receiv: Permite recibir archivos en forma hexadecimal desde el computador.

Xmit: Permite transmitir archivos tipo texto

rxBin: Permite recibir archivos tipo binario.

rxTxt: Permite recibir archivos tipo texto.

reM: No está disponible.

- Escoja la opción **Conf**, el siguiente menú aparecerá en el despliegue

Modo [1] [2] [3]

luego escoja la opción [1] que indica modo de transmisión.

El modo de transmisión le indica al microcontrolador si hay bit start, bit de parada y la extensión de la palabra a recibir o transmitir.

Después de seleccionar el modo, se debe escoger la velocidad de transmisión. Escoja la opción 96 la cual equivale a 9600 Baudios en el menú que aparece a continuación:

[03] [06] [12] [24]
[48] [**96**] [19.2]

- Luego presione la tecla escape (**ESC**) para salirse al menú de comunicaciones. En éste menú, digite la tecla X (**Xmit**) y de esta forma transmitirá el archivo que inicialmente había digitado en el Microlab 51 al computador. Ahora podrá grabar el programa en un archivo, y de esta forma usarlo en cualquier otro momento.

Nota: Si desea mayor información sobre el manejo del Microlab 51, diríjase al manual del usuario versión 1.0

4.1.3.5 Manejo de Software Adicional para Análisis de Programas

Debido a las limitaciones que tiene el sistema de desarrollo Microlab 51 (básicamente limitaciones de memoria), en algunas aplicaciones resulta más conveniente digitar los programas en un computador y luego enviarlos hacia el sistema de desarrollo en forma hexadecimal, lo cual resulta conveniente porque éste no necesita interpretar o compilar las instrucciones de los programas realizados en ensamblador, sino que simplemente ejecuta las instrucciones que se encuentran en su memoria.

Utilice el archivo generado en el ejercicio anterior, cargándolo en el editor de texto del **MICROWIN**.

Para ensamblar el programa utilice la opción **Ensamblar** en el menú principal del **MICROWIN**, luego escoja la opción **Ensamblador** y dentro de ésta **Ensamble MCS51.** Después de compilar y generar el archivo hexadecimal, envíelo hacia el Microlab 51 utilizando la opción **Comunicaciones**, **Enviar Archivo Hexadecimal** y ejecute el programa.

Ejercicio Propuesto

Compare los procedimientos anteriores (digitando el programa directamente en el editor del Microlab o utilizando el computador). Qué ventajas tiene uno con respecto al otro.

Escoja la opción **Simular** luego la opcion **Simulador MCS51** del MICROWIN. Escoja el tipo de microcontrolador a usar, en este caso el 8031, en la pantalla del computador se mostrará todos los componentes del microcontrolador, registros, puertos, etc. Carge el programa INTEL.HEX y corralo. Para el manejo del Simulador refiérase al manual de usuario.

Investigue que herramientas de desarrollo hay disponibles para los microcontroladores en el mercado.

4.2. PRACTICA No 2: MANEJO DE PUERTO Y RETARDOS DE TIEMPO

4.2.1. Objetivo General

Instruir al usuario en el manejo de los puertos del microcontrolador, para el control de periféricos y la visualización de resultados generados por retardos de tiempo

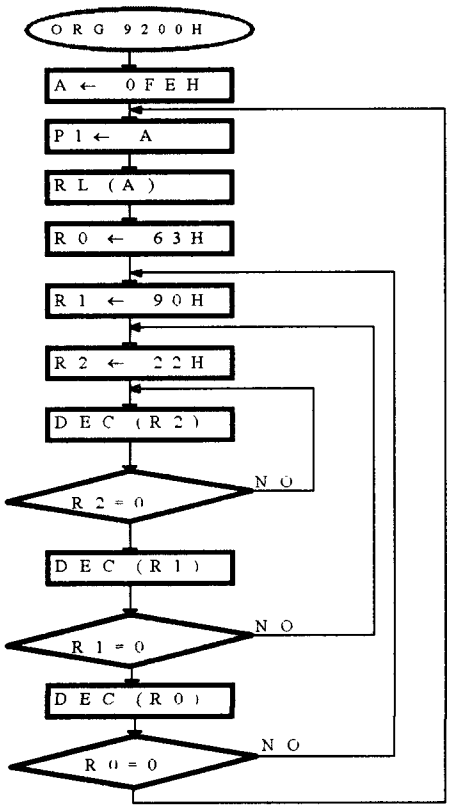
4.2.2. Objetivos Específicos

- Conocer la importancia que tiene generar un retardo de tiempo en una aplicación o proceso determinado.
- Realizar retardos por software.
- Analizar el tratamiento de rutinas.
- Visualizar resultados a través del Puerto 1 (P1).
- Realizar el movimiento de tablas en memoria externa.
- Mostrar los resultados de una tabla en un despliegue 7 segmentos de ánodo común.

4.2.3. Primera Parte: Manejo de Retardos

Realizar un programa que produzca el desplazamiento de un bit a la izquierda en el Acumulador, el cual se visualice por medio de leds a través del Puerto P1. La velocidad de desplazamiento del bit es de un segundo y dependerá de los valores que se le asignen a los registros R0, R1 y R2. El decremento sucesivo de estos valores de una forma anidada producen este retardo de tiempo.

El siguiente es el diagrama de flujo que produce el desplazamiento y genera el retardo



Procedimiento

Conecte en el puerto P1 ocho leds con sus respectivas resistencias limitadoras de corriente (220Ω) haciendo una conexión en ánodo común (ver figura 4.1).

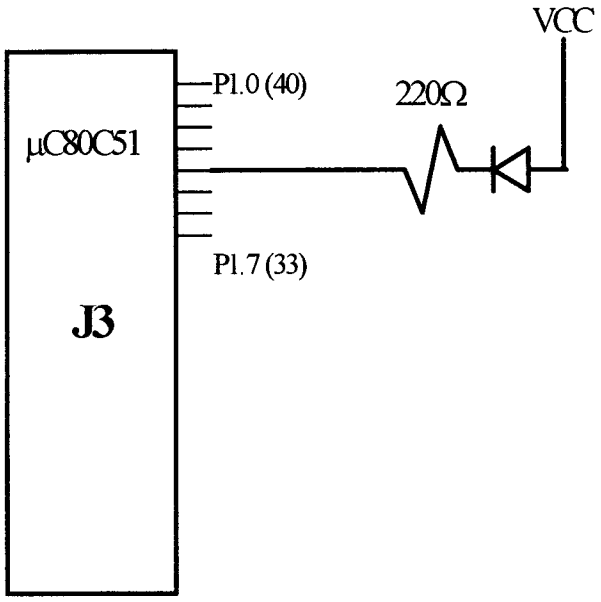


Figura. No 4.1 Conexión de los leds al Microcontrolador

Digite el siguiente programa para ser ejecutado en el Microlab-51:

```
                ORG     9200H   ; Origen del programa en la posición 9200H
                MOV     A,#0FEH; ACC ← 0FEH, Cargar ACC con FEH
Ciclo           MOV     P1,A    ; Cargar P1 con el contenido del ACC
                RL      A       ; Rotación de un bit a la izquierda
RETARDO         MOV     R0,#63H         ; Cargamos el Registro 0 (R0) con #63H
CicloA          MOV     R1,#90H         ; Cargamos el Registro 1 (R1) con #90H
CicloB          MOV     R2,#22H         ; Cargamos el Registro 2 (R2) con #22H
CicloC          DJNZ    R2,CicloC       ; Decrementa R2 hasta que R2=0
                DJNZ    R1,CicloB       ; Decrementa R1 hasta que R1=0 y salta
                                        ; nuevamente al CicloB. Por cada lazo de
                                        ; R1, ejecuta 22H veces el lazo R2.
                DJNZ    R0,CicloA       ;Lo mismo que las instrucciones
```

```
                        ;anteriores, cada vez que decrementa R0,
                        ,decrementa 90H  veces a R1 y 90H*22H
                        ;veces a R2
JMP     Ciclo           ;Comienza de nuevo el ciclo
                        ;indefinidamente.
```

Observe el resultado obtenido y explique de que manera se produce el retardo de un segundo

Ejercicio Propuesto

Realice un programa que desplace un bit a la derecha en el Acumulador y se visualice en un arreglo de 7 leds conectados en anodo comun, a través del puerto P1

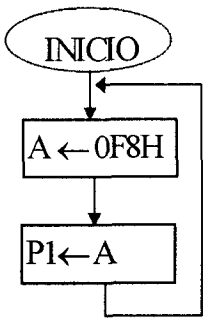
4.2.4. Segunda Parte: Manejo de Despliegue

Procedimiento

Conecte un despliegue 7 segmentos de ánodo común a la salida del puerto P1. Las conexiones del puerto al despliegue deben conservar el siguiente orden: pin 1.0 con el segmento A, pin 1.1 con el segmento B y así sucesivamente.

Para visualizar un número en el despliegue 7 segmentos, se debe enviar al puerto el código correspondiente al número deseado, por ejemplo: para visualizar el número

siete (7) se deben encender los segmentos a, b y c respectivamente los cuales corresponden al número #0F8H en hexadecimal. A continuación aparece el diagrama de flujo que genera en el despliegue 7 segmentos el número siete (7):



Ejercicio Propuesto

Realice el programa, verifique su funcionamiento y luego pruebe con números diferentes de siete (7) entre cero (0) y nueve (9).

4.2.5. Tercera Parte: Manejo de Tablas

Procedimiento

Genere una tabla con el contenido de los códigos en hexadecimal de los números del 0 al 9 y carguela en memoria externa.

El siguiente ejemplo muestra la forma de elaborar la tabla:

```
TABLA MOV   DPH,#93H      ;Direccionamos el DTPR
      MOV   DPL,#00H      ; a la posición 9300H
      MOV   A,#0C0H       ; ACC ← 0
      MOVX  @DPTR,A       ; DPTR ← 0
      INC   DPTR          ; DPTR = 9300H + 1
      MOV   A,#0FCH       ; ACC ← 1
```

Continúe el mismo procedimiento hasta llegar al número 9

```
      MOVX  @DPTR,A       ; DPTR ← 9
      RET
```

Realice un programa que produzca un retardo de tiempo de 2 segundos utilice como guía el procedimiento de la primera parte.

Digite el siguiente programa para ensamblar en el Microlab-51

```
      ORG   9200H         ; Origen del programa en la posición 9200H

      ACALL TABLA         ; Cargar tabla en la memoria

PASOA MOV   DPTR,#92FFH   ; Inicializar DPTR en 92FFH
      MOV   A,DPL         ; ACC ← DPL.. DPL=FFH
PASOB INC   A             ; ACC ← FFH+01H
      MOV   DPL,A         ;DPL ← 00H
      MOV   DPH,#93H      ; DPH ← 93H
      MOVX  A,@DPTR       ;ACC ← 9300H
      MOV   P1,A          , Visualización del contenido del DPTR
                          ; en el puerto P1
      ACALL RETARDO       ; Llamada a subrutina para retardo de 2 Sg.
      MOV   A,DPL         ; ACC ← 00H
      CJNE  A,#09H,PASOB  ; Si ACC = 09H, saltamos a PASOA, de lo
                          ;contrario saltamos a PASOB
      LJMP  PASOA         ; Continuamos el ciclo indefinidamente

RETARDO
      ;Escribir aquí el programa de retardo de tiempo para 2 segundos
      RET
```

```
TABLA
        ;Escribir aquí la tabla que se explico anteriormente
        RET
        END                              ; final del programa
```

Nota: Tenga presente que el llamado a una subrutina se hace con la instrucción **ACALL** [SUBRUTINA] y al final de la subrutina se escribe la instrucción **RET** indicando así su finalización.

Ejercicio Propuesto

Complete el programa anterior de tal manera que la visualización de los números se realice en forma ascendente y descendente.

4.3. PRÁCTICA 3:TEMPORIZADORES E INTERRUPCIONES

4.3.1. Objetivo General

Instruír al usuario mediante un ejemplo de despliegue el manejo de temporizaciones e interrupciones.

4.3.2. Objetivos Específicos

- Instruir al usuario en la programación de los Timer
- Programar el registro TMOD y TCON.
- Manejar el registro de habilitación de las interrupciones (IE) y el registro de niveles de prioridad de las interrupciones (IP).
- Instruir al usuario en el direccionamiento de las interrupciones, en la tabla de vectorizaciones.
- Analizar como conseguir tiempos reales en las temporizaciones
- Trabajar en tiempo real las temporizaciones.

4.3.2.1. Primera Parte:Utilización de Decodificador BCD a 7 Segmentos

Procedimiento

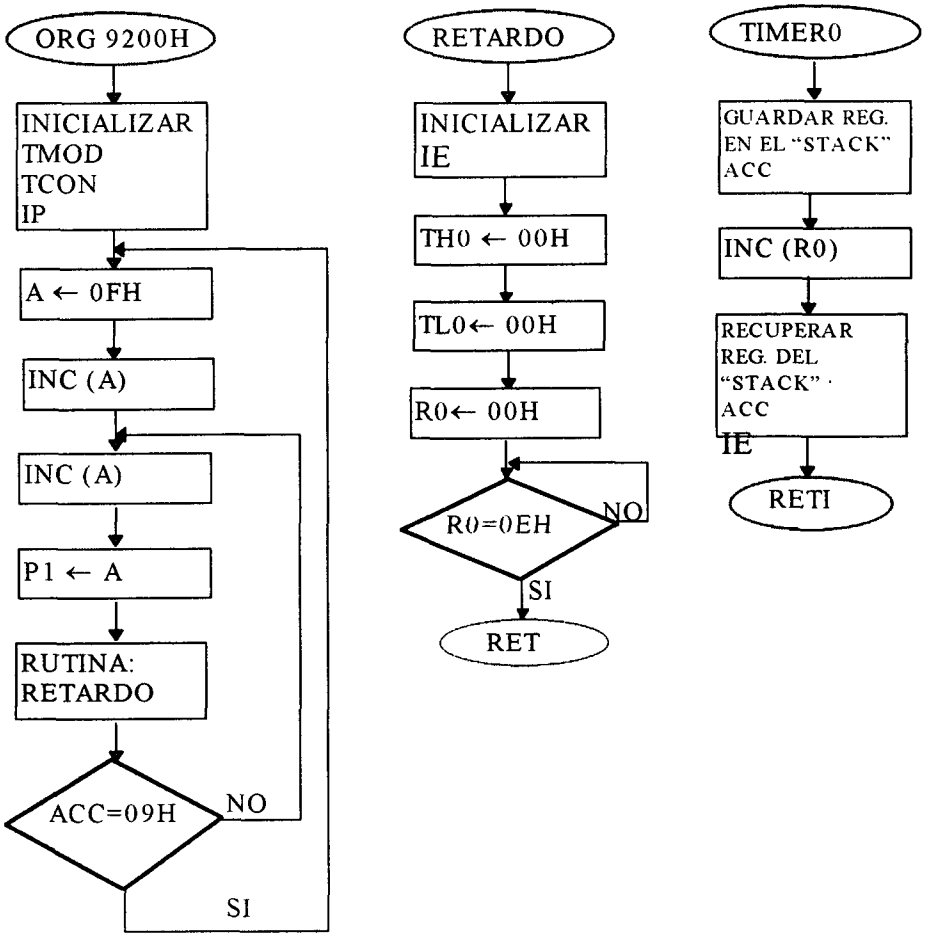
Realize un programa en ensamblador el cual genere un contador cíclico de 0 a 9 con un retardo entre números de 1 segundo.

Conecte en el puerto P1 un decodificador BCD a 7 segmentos (7447), y a éste conectarle un despliegue 7 segmentos de ánodo común para la visualización de los números de 0 al 9.

Para programar el Timer y la interrupción, debe tener en cuenta lo siguiente:

- Realice el procedimiento que genere los números del 0 al 9
- Programe el registro TMOD para que habilite el Timer 0 en modo 1 (16 bits) como temporizador y así tener el control por software.
- Programe el registro TCON para habilitar el temporizador/contador 0
- Programe el registro IE para que habilite la interrupción del Timer 0.
- Programe el registro IP para tener alta prioridad en la interrupción del Timer 0.
- Utilice la interrupción para hacer el conteo de 1 segundo.

El siguiente diagrama de flujo es el procedimiento que debe seguirse para la ejecución del programa:



Digite el siguiente programa

```
        ORG     9200H
CICLOA  MOV     A,#0FH
CICLOB  INC     A
        MOV     P1,A
        ACALL   RETARDO
        CJNE    A,#19H,CICLOB
```

```
                JMP     CICLOA
******* TRATAMIENTO DE LA INTERRUPCIÓN DEL TIMER 0 *******

TIMO            MOV     TH0,#$00        ; Para conseguir la máxima
                MOV     TL0,#$00        ; Temporización
                INC     R0              ; R0 ← R0+1
                RETI                    ;Retornamos de la interrupción

******* RUTINA DE RETARDO *******

RETARDO
                MOV     R0,#00H
                MOV     TCON,#00H
                MOV     TMOD,#$01
                MOV     IP,#02H
                MOV     IE,#82H
                MOV     TCON,#10H
CICLO           CJNE    R0,#0EH,CICLO
                RET

*******  LLAMADA A LA INTERRUPCIÓN  *******
                ORG     9FF9H
                LJMP    TIMO
                END.
```

Explique cómo se produce el retardo de tiempo.

De qué otra forma se puede generar este retardo de tiempo?

Explique que indica 0EH en la instrucción CICLO CJNE R0,**#0EH**,CICLO

Ejercicio Propuesto

- Repetir la práctica anterior sin utilizar el decodificador BCD a 7 segmentos (software únicamente) y además emplee el Timer 1 trabajando en modo 0 (13 bits).

Cambie el retardo de tiempo a 5 segundos

4.4. PRACTICA 4: RECEPCIÓN Y TRANSMISIÓN POR EL PUERTO SERIE

4.4.1. Objetivo General

Orientar al usuario en la utilización del puerto serial de manera tal que permita una correcta y confiable transmisión y/o recepción de la información.

4.4.2. Objetivos Específicos

- Transmitir por el puerto serie bytes de datos capturados o recibidos por la línea RXD.
- Recibir por el puerto serie bytes de datos capturados o recibidos por la línea TXD.
- Programar el registro de control del puerto serial (SCON).
- Programar la pantalla de cristal liquido (LCD)[2] dispuesta en el sistema de desarrollo del Microlab-51

[2]Para la correcta programación del LCD refirirse al Apéndice E (Características del Display) del manual de usuario del Microlab-51.

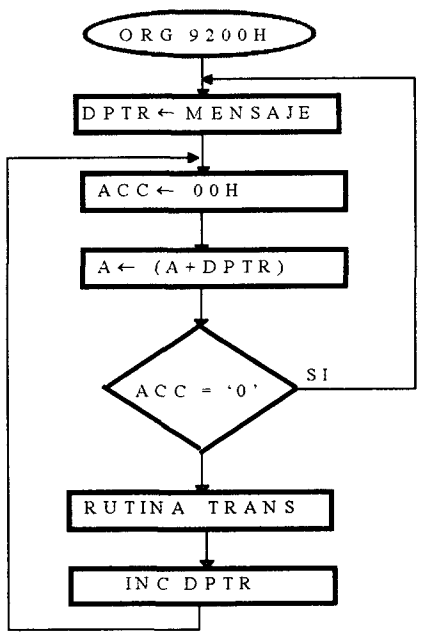
4.4.3. Primera Parte: Transmisión por el puerto serie

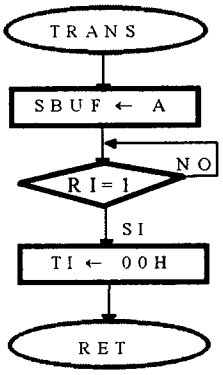
Procedimiento

- Conecte el computador con el Microlab 51 a través del puerto serie.
- Transmita por el puerto serie un mensaje con el nombre de la "Universidad Autónoma", el mensaje debe ser visualizado en la ventana de comunicaciones MICROWIN- COMUNICACIONES.

- Cargue el mensaje a transmitir en el DPTR.
- Ejecute la instrucción para la lectura de tablas en memoria externa MOVC.
- Programe el registro SCON para que habilite la bandera de interrupción en la transmisión (TI) y realice la subrutina que transmita el mensaje.

A continuación se muestra el diagrama de flujo para generar la transmisión del mensaje:





- Digite el siguiente programa

```
                ORG   9200H
TxMssg mov      DPTR,#mensaje       ;Cargar el DPTR con el mensaje
Ciclo           CLR   A
                MOVC  A,@A+DPTR     ;Cargar ACC con el contenido de DPTR
                CJNE  A,#'\0',TxCh  ;
                JMP   TxMssg

TxCh            CALL  Trans         ;Llamada a la rutina de transmisión
                INC   DPTR
                JMP   Ciclo

***** Subrutina transmisión de mensaje *****

Trans           MOV   SBUF,A:El contenido de ACC se envía al SBUF
                JNB   TI,$    ,Permite la transmisión del mensaje
                CLR   TI      ,Interrumpimos la transmisión
                RET
```

```
***** Mensaje a transmitir *****
mensaje DB      'Universidad Autónoma\r\n\0'
                END
```

4.4.4. Segunda Parte: Recepción por el Puerto Serie

Procedimiento

- Realice un programa que comunique directamente el computador con el Microlab 51.
- Muestre en el LCD cada uno de los caracteres generados al pulsar una tecla del computador.

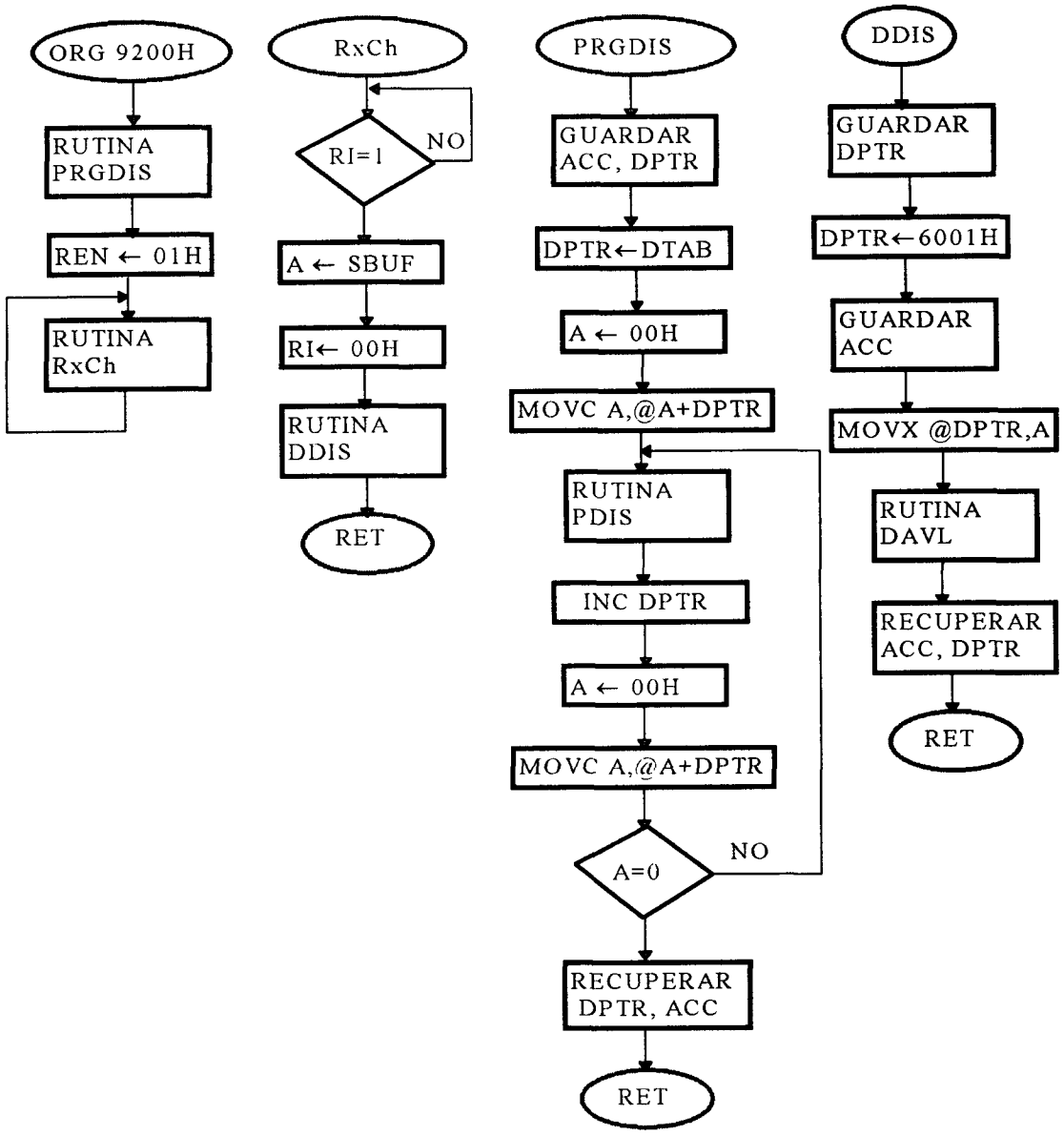
Programe las direcciones de control y operación del LCD.

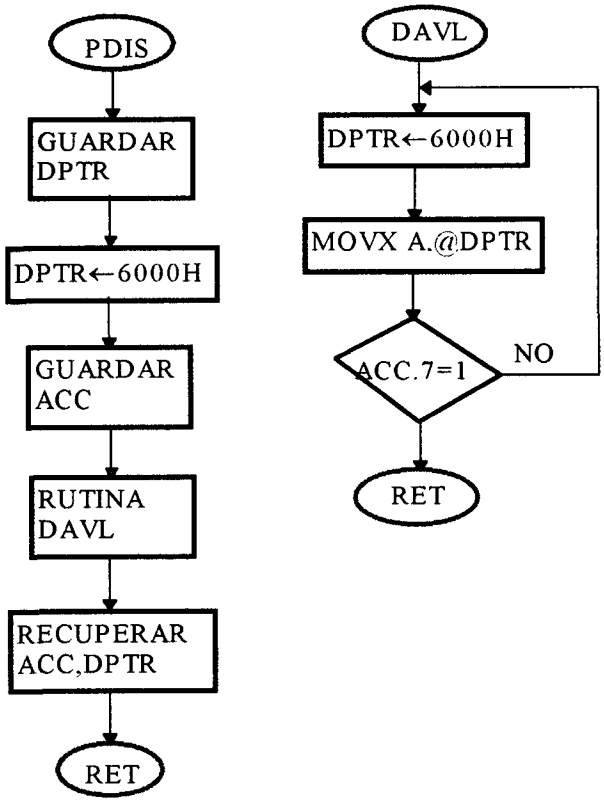
Nota:

Las subrutinas de programación del display se encuentran en las páginas 46, 47 y 91 del manual de usuario del Microlab-51, utilizando estas subrutinas se puede visualizar en el LCD los caracteres generados en el teclado del computador. Las direcciones de control y operación son la 6000H y 6001H respectivamente.

Programe el registro SCON.4 (REN) para permitir la recepción de caracteres.

El diagrama de flujo del programa es el siguiente:





El programa en ensamblador es el siguiente:

```
                ORG     $9200
                CALL    PRGDIS
                SETB    REN     ;Permite la recepción
TxMssg          CALL    RxCh
                JMP     TxMssg ;Se repite el ciclo

***** Subrutina Recepción de Carácter
RxCh            JNB     ri,$    ; Si RI=1 permite la recepción
                MOV     A,sbuf  ;El contenido de SBUF se guarda en ACC
                CLR     RI
                CALL    ddis
                RET

;***** CONTROL DE DISPLAY *****
PRGDIS          PUSH    ACC
                PUSH    DPH
                PUSH    DPL
                MOV     DPTR,#DTAB    ;Cargamos la tabla en el DPTR
```

```
                CLR     A
                MOVC    A,@A+DPTR       ;Traer el primer código
LOOP            CALL    PDIS            ;Envía palabra de control al display
                INC     DPTR
                CLR     A
                MOVC    A,@A+DPTR       ;Trae la sgte. palabra de control
                CJNE    A,#00H,LOOP     ;repite hasta encontrar 00H.
                POP     DPL
                POP     DPH
                POP     ACC
                RET

;***** Tabla para programar el display del Microlab-51
DTAB            DB      38H     ;Control con 8 bits
                DB      0FH     ;parpadeo del cursor rectangular, visualización
                                ;del cursor, enciende el display.
                DB      06H     ;Incremento del registro AC (Address Counter)
                                ;al escribir.
                DB      01H     ;Limpiar el display
                DB      80H     ;Posicionar el display en el comienzo de la
                                ;primera línea.
                DB      00H     ;Marca el final de la tabla.

;***** PROGRAMACIÓN DEL DISPLAY *****
PDIS            PUSH    DPH
                PUSH    DPL
                MOV     DPTR,#6000H     ;Apunta al registro de control del
                                        ;display.
                PUSH    ACC
                MOVX    @DPTR,A         ;Envia palabra de control
                LCALL   DAVL
                POP     ACC
                POP     DPL
                POP     DPH
                RET

;***** Subrutina que espera hasta que el LCD este desocupado *****
DAVL            MOV     DPTR,#6000H     ;Apunta al registro de control del
                                        ;display.
                MOVX    A,@DPTR         ;Trae palabra de control
                JB      ACC.7,DAVL      ; Pregunta si puede enviar nuevo
                                        ;dato.
                RET
```

```
;***** Subrutina para escribir en el LCD *****
DDIS        PUSH  DPH
            PUSH  DPL
            MOV   DPTR,#6001H   ;Apunta hacia el LCD
            PUSH  ACC
            MOVX  @DPTR,A       ;Envia carácter
            LCALL DAVL
            POP   ACC
            POP   DPL
            POP   DPH
            RET
            END
```

Ejercicio Propuesto

Transmisión y Recepción por el Puerto Serie

- Realize un programa que permita visualizar en el LCD del Microlab 51 y en la pantalla del computador (ventana de comunicaciones) los caracteres digitados en el teclado del computador.
- Realize un programa que permita visualizar en el LCD del Microlab 51 y en la pantalla del computador los caracteres digitados en el teclado del Microlab 51.

APLICACIONES PRÁCTICAS CON MICROCONTROLADORES (8051)

A continuación se sugieren una serie de prácticas que podríamos calificar como proyectos de variada dificultad. A pesar de que algunos de ellos requieren una circuiteria ésta no es demasiado compleja de construir.

1. Comunicación entre dos sistemas de desarrollo. (Microlab-51)
2. Control de una pantalla de cristal líquido.
3. Regulación de luminosidad (Regulación controlada de carga resistiva).
4. Control de un teclado hexadecimal.
5. Generación de una señal PWM.
6. Letrero publicitario.
7. Un reloj digital

Se anima al usuario a que intente llevar a término estos proyectos, con objeto de que adquiera experiencia en el diseño con microcontroladores de la familia 8051, y dado a que la práctica es base fundamental para que ésta experiencia pueda adquirirse.

CONCLUSIONES

Creada la necesidad de hacer más versátil la programación y depuración de las aplicaciones con microcontroladores, se vio la necesidad de desarrollar un software para comunicaciones entre sistemas de desarrollo con microcontroladores y computadores, diseñado por nosotros; se aspira que en corto tiempo se convierta en una valiosa herramienta para las prácticas de laboratorio, proyectos de investigación, tesis de grado y aplicaciones especificas, entre otras.

Con el sistema de desarrollo Microlab-51 se cuenta con una herramienta interesante y útil para personas que se inician en el aprendizaje de los microcontroladores, ya que el usuario puede analizar de una manera más sencilla la arquitectura, el uso del conjunto de instrucciones, manejo y control de puertos, manejo de interrupciones y el manejo de las comunicaciones entre otras.

Se encontró que el ensamblador del Microlab-51 interpreta como errores de sintaxis programas que contengan texto escrito en minúsculas, como también, registros de

funciones especiales los cuales son direccionables bit a bit que no reconoce. Esto debido a que el Microlab-51 por ser un sistema desarrollado para aprendizaje, no habilitó esas utilidades por disposiciones limitadas de memoria. Dichos inconvenientes se solucionan si se editan y ensamblan los programas en el software MICROWIN y posteriormente se transmiten, en formato hexadecimal hacia el Sistema de Desarrollo por el puerto serial.

MICROWIN además de ser una herramienta para el área de microcontroladores en Electrónica, puede ser utilizado para la materia de microprocesadores I y II en Ingeniería Eléctrica, por lo tanto se hará más fácil la implementación de prácticas con el **MICROPROFESSOR**.

En ingeniería como en otras ramas, el tiempo juega un papel primordial y más cuando se presentan problemas que requieren de una pronta solución, nuestro trabajo de ingeniería fue automatizar y reducir el tiempo en la elaboración y ejecución de los programas.

MICROWIN además de servir para estos sistemas de desarrollo sirve para cualquier sistema que utilice una comunicación serial como interfaz. Con lo cual, la universidad Autónoma queda con un poderoso software para que la persona que lo utilice le saque el mejor provecho posible.

RECOMENDACIONES

Si se necesita desarrollar una aplicación específica, es conveniente crear su propio sistema de desarrollo, ya que de esta forma se podrá tener a disposición todas las utilidades del microcontrolador.

Se sugiere adicionar en el pensum de eléctrica una materia en la cual se enseñe microcontroladores, esta sería un buen complemento al área de control y digitales.

Para alargar más la vida útil del microcontrolador evitando su deterioro, se recomienda trabajar con transistores a la salida de los puertos del microcontrolador, para que manejen la demanda de corriente del circuito. Para las prácticas de laboratorio de este trabajo de grado, puede utilizar el circuito integrado ULN2803 que contiene 8 transistores que se conectan directamente al puerto del microcontrolador

REFERENCIAS BIBLIOGRÁFICAS

ANGULO, J María. Enciclopedia de electrónica moderna. Madrid : Paraninfo s.a. 1990

BYTE. Artículos Circuit's Ceelar Ink.

CARRERAS GARCÍA, Juan. El 8051: Un microcontrolador prolífico. En: Revista española de electrónica. No 463(Jun 1993); P 46-54

CIARCIA, Steven. Why microcontroller. In: Byte. Vol 13 No 8 (Ago./sep. 1988)

--------. Computers on the Brain. In: Byte. Vol 13 No 6 (Jun./Jul. 1988)

--------. Build the circuit Cellar IC Tested. In: Byte. Vol 12 No 13 (Nov de 1987) ; P 303-313

--------. --------. In: Byte. Vol 12 No 14 (Dic de 1987); P 283-289

--------. Build the Basic-52 Computer Controller. In: Byte (Aug de 1985); P 105-117

DAURA LUNA, Francesc. Microcontrolador RISC de bajo coste. En: Revista mundo electrónico. No 218 (Jun 1991); P 80-84

--------. PIC16CXX: una nueva familia de microcontroladores RISC CMOS de altas prestaciones/y 2. En: Revista española de electrónica. No 486 (mayo 1995); P 54-59

DINWIDDIE, George. An 8031 In-Circuit Emulator. In: Byte (Jul de 1986); P 181-194

GONZÁLEZ VÁZQUEZ, José Adolfo. Introducción a los microcontroladores. Madrid : McGRAW-HILL, 1992

GONZÁLEZ VÁZQUEZ ,J. Adolfo; CABEZA SOBERON M. Lourdes y MARTÍNEZ DÍAZ E. Introducción a los Microcontroladores de 16 Bits. Madrid : McGRAW-HILL, 1994

GONZÁLEZ VÁZQUEZ, José Adolfo. El procesador booleno del microcontrolador 8051. En: Revista española de electrónica. No 471 (Feb 1994); P 56-63

IDIONDO, Roberto y ANGULO, José María. Diseño con microcontroladores monochip. En: Revista española de electrónica. No 463(Jun 1993); P 40-45

INTEL Corporation. Handbook Embeebed Microcontrollers. 1981

LONDOÑO POSADA, Jorge Mario. Periféricos e interfaces. Colombia : Universidad pontificia bolivariana 1995

MARTÍNEZ PÉREZ, Javier y BARRON RUIZ, Mariano. Prácticas con microcontroladores. Madrid : McGRAW-HILL 1993

Manual de usuario de Microlab-51. Ver 1.0

NATIONAL SEMICONDUCTORES. Linear applications Vol 1. Santa Clara 1980.

ROMERAL MARTÍNEZ, José Luis y BALCELLS SENDRA, José. Herramientas de emulación. En: Revista española de electrónica. No 463 (Jun de 1993); P 32-38

SEARGENT & SCHOEMAKER. Interfacing microcomputers to the Real Word. Addison Wesley Publishing Co., 1981.

SOLUTIONS. January/February, 1983. Application Note: Implementing Automotive deshboard Functions with the 8051 microcontrollers.

--------. May/June 1986. microcontrollers: New extensions for new niches.

ANEXO I

SOFTWARE PARA COMUNICACIONES ENTRE UN SISTEMA DE DESARROLLO Y EL COMPUTADOR

1.1. INTRODUCCIÓN

El presente anexo contiene el Manual de Usuario explicatorio del software **MICROWIN**, complemento de la tesis "**Desarrollo de Prácticas de Laboratorio con Microcontroladores para Docencia Universitaria**", que como su nombre lo indica, explica o enseña cómo se maneja el software de esta tesis.

MICROWIN es una aplicación de soporte para comunicaciones, con la cual se pueden llevar a cabo transmisiones/recepciones de archivos tipo texto (archivos con extensión .TXT) hexadecimales (extensión .HEX) de tal manera que el usuario tenga varias opciones para trabajar con un Sistema de Desarrollo[1]. **MICROWIN** es una aplicación desarrollada en ambiente Windows, por lo que requiere de éste Software

[1]El Sistema de Desarrollo hace referencia al Microlab-51 o Micro-Proffesor, el cual debe contar con el respectivo puerto serial para comunicarse con el computador

para su funcionamiento. Las exigencias en cuanto a memoria y tipo de computador son las mismas que requiere **Microsoft Windows.**

Las funciones importantes de esta aplicación son:

- Permite la edición de archivos tipo texto, siendo posible crear los programas en lenguaje ensamblador, y luego transmitirlos a un Sistema de Desarrollo como, por ejemplo, el Microlab-51.

- **MICROWIN** también ofrece al usuario todas las facilidades en el ensamble, enlace (Linking) del programa fuente, para producir un archivo hexadecimal y transmitirlo al Sistema de desarrollo para realizar la depuración de éste.

- Para simular los programas fuente, **MICROWIN**, dispone de los Simuladores[2] AVSIM51 y AVSIMZ80 para las Familias MCS-51 y Z80 respectivamente.

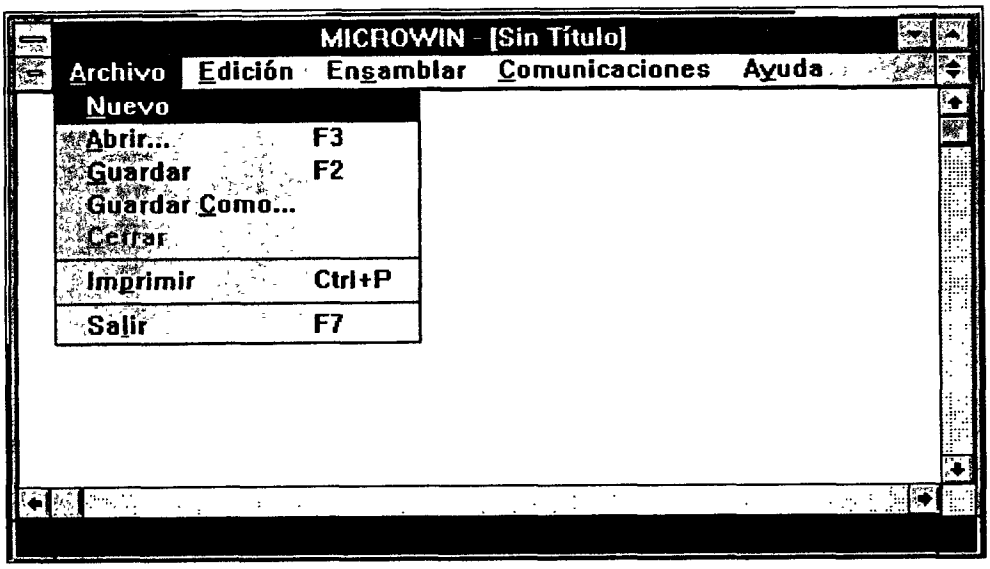
[2]Estos simuladores son productos de la compañía AVOCET.

1.2. MENÚ Y TECLAS DE FUNCIONES

El menú horizontal que aparece en la parte superior de la pantalla se puede accesar presionando **<Alt> + <Tecla Subrayada>**, utilizando el Ratón (Mouse), o utilizando el conjunto de teclas abreviadas para tal fin.

A continuación se describe cada uno de los comandos que aparecen en el menú horizontal.

1.2.1. Comandos del menú Archivo



Nuevo

Permite abrir y crear un nuevo documento. Cuando elija Nuevo, se ofrecerá la

posibilidad de guardar las modificaciones que haya introducido en el documento en el que esté trabajando.

Abrir

Abre un documento que ya existe, permite hacer las modificaciones del caso y salvarlo mediante la opción Guardar (F2).

Se puede acceder a esta función presionando la tecla abreviada F3.

Para abrir rápidamente uno de los últimos documentos con los que haya trabajado, elija su nombre de la parte inferior del menú **Archivo.**

Guardar

Salva o Guarda las modificaciones introducidas en el documento en el cual se ha estado trabajando. Si es la primera vez que guarda el documento, se mostrará el cuadro de diálogo **Guardar como** donde podrá escribir un nombre para el documento. Cuando elija Guardar, el documento permanecerá abierto, por lo que podrá seguir trabajando en él

Se puede acceder a esta función presionando la tecla **F2**.

Guardar como

Guarda un documento nuevo o uno ya existente.

Permite guardar un nuevo documento en formato tipo texto (extensión .TXT, . BAT u otra extensión definida por el usuario). Puede asignarle un nombre a un documento nuevo o guardar un documento ya existente con otro nombre (si grabó un archivo existente con otro nombre, el documento original permanecerá sin cambios). Cuando elija **Guardar como**, el documento permanecerá abierto, por lo que podrá seguir trabajando en él.

Imprimir

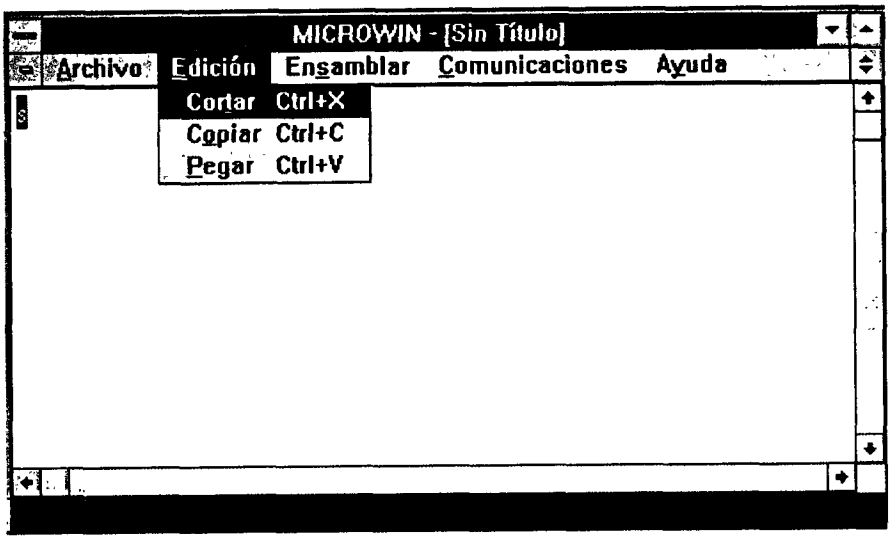
Imprime el documento en el cual se está trabajando, también se puede imprimir apartes del texto seleccionado previamente. Si no hay texto en el portapapeles para imprimir, aparecerá un mensaje indicando que no hay texto para imprimir.

Se puede acceder a esta función presionando las teclas **<CTRL>** + **<P>**.

Salir

Cierra el documento con el cual se esté trabajando, dando por terminada la aplicación **MICROWIN**. El sistema ofrece la posibilidad de guardar el documento antes de salir.

Se puede acceder a esta función presionando la tecla **F7**.



1.2.2. Comandos del menú Edición

Cortar

Borra texto seleccionado de un documento y los guarda en el Portapapeles, sustituyendo el contenido anterior del mismo. Este comando sólo estará disponible cuando haya texto seleccionado.

Se puede acceder a esta función presionando las teclas **<CTRL> + <X>**

Copiar

Copia texto seleccionado del documento y lo guarda en el Portapapeles, dejando intacto el original y sustituyendo el contenido anterior del Portapapeles. Este comando sólo estará disponible cuando haya texto seleccionado.

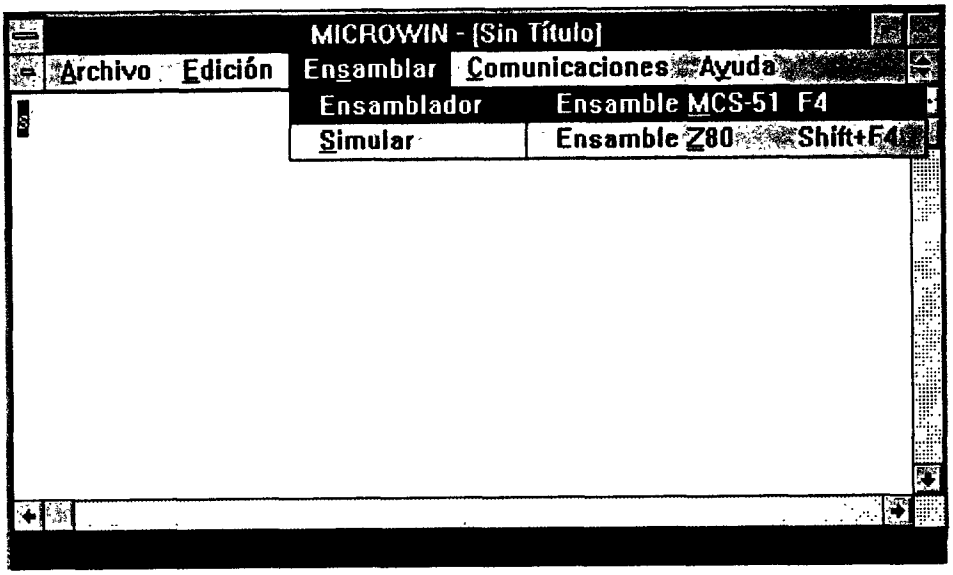
Se puede acceder a esta función presionando las teclas **<CTRL> + <C>**

Pegar

Incrusta o copia en el archivo la información contenida en el Portapapeles. **MICROWIN** sitúa dicha información en la posición actual del punto de inserción. Este comando estará disponible cuando se haya copiado información o cuando se haya cortado y guardado en el Portapapeles.

Se puede acceder a esta función presionando las teclas **<CTRL> + <V>**

1.2.3. Comandos del menú Ensamblar



Ensamblador

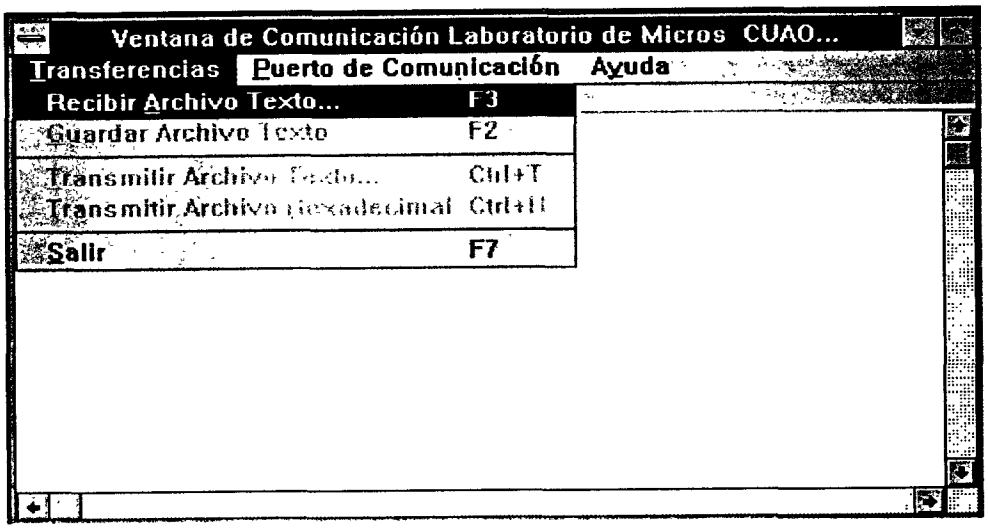
Con esta opción se puede seleccionar el ensamble para los microcontroladores de la **Familia 80C51** o, por el contrario, ensamblar el programa fuente para los microprocesadores de la **Familia Zilog Z80**.

Para acceder al ensamble de la Familia 80C51 utilice la tecla **<F4>**. Para acceder al ensamble de la Familia Zilog Z80 utilice las teclas **<Shift>** + **<F4>**. Los archivos previamente ensamblados son salvados o guardados con los nombre **Intel.Hex** y **Zilog.Hex** respectivamente.

Simular

Permite Simular los programas fuente para las familias de microprocesadores Z80 y microcontroladores de la familia 80C51. Para más información acerca del manejo de estos simuladores, debe referirse a los manuales del usuario de la compañía AVOCET.

1.2.4. Comando del Menú Comunicaciones



Comunicaciones

Esta opción es la más importante para el trabajo con el **MICROWIN**, pues a través de ella se pueden llevar a cabo todas las funciones de transferencia de archivos entre el computador y el Sistema de Desarrollo o viceversa.

Esta opción permite cambiar a otro ambiente o ventana. Los comandos de este menú se presentan a continuación:

1.2.4.1. Comandos del Menú Transferencias

Recibir Archivo Texto

Recibe un archivo de texto procedente del Sistema de Desarrollo. Para realizar esta operación se debe seguir los siguientes pasos

1. Configurar el puerto de comunicaciones.
2. Abrir el puerto de comunicaciones.
3. En el menú transferencias, elija recibir archivo texto.
4. En las listas "Unidades" y "Directorios", elija la unidad y el directorio donde desee almacenar el nuevo archivo.
5. En la lista "Nombre del archivo" escriba el nombre del archivo.
6. Elija el botón aceptar.
7. Envíe el archivo desde el Sistema de Desarrollo.
8. Guarde el archivo.

Puede utilizar la tecla **F3** para accesar a esta opción.

Guardar Archivo Texto

Guardar o salva el archivo enviado por el Sistema de Desarrollo.

Puede utilizar la tecla **F2** para accesar a esta opción.

Transmitir Archivo Texto

Transmite al Sistema de Desarrollo un archivo tipo texto a través del puerto serial. Al ejecutar esta instrucción, se debe tener en cuenta:

1. Configurar el puerto de comunicaciones.
2. Abrir el puerto de comunicaciones.
3. En el menú transferencias, elija transmitir archivo texto.
4. En las listas "Unidades" y "Directorios", elija la unidad y el directorio donde se encuentra el archivo que desee enviar.
5. En la lista **Nombre del archivo** elija el archivo que desee enviar o bien escriba el nombre del mismo
6. Elija el botón aceptar.

Puede utilizar las teclas **<CTRL> + <T>** para accesar a esta opción.

Nota: Si utilizó la opción ensamble MCS-51, del menú ensamblar, el archivo que

desea enviar aparecerá por defecto con el nombre y ruta como: **"C:\Microwin\Avc51\Intel.Txt".** Lo mismo ocurre si utiliza la opción ensamble Z80, cuya ruta y nombre serán: **"C:\Microwin\Avcz80\Zilog.Txt"**

Transmitir Archivo Hexadecimal

Transmite al Sistema de Desarrollo un archivo con formato hexadecimal a través del puerto serie. Al ejecutar esta instrucción, se debe tener en cuenta:

1. Configurar el puerto de comunicaciones.
2. Abrir el puerto de comunicaciones.
3. En el menú transferencias, elija transmitir archivo hexadecimal.
4. En las listas "Unidades" y "Directorios", elija la unidad y el directorio donde se encuentra el archivo que desee enviar.
5. En la lista "Nombre del archivo" elija el archivo que desee enviar o bien escriba el nombre del mismo
6. Elija el botón aceptar.

Puede utilizar las teclas **<CTRL> + <H>** para accesar a esta opción.

Nota: Si utilizó la opción ensamble MCS-51, del menú ensamblar, el archivo que desea enviar aparecerá por defecto con el nombre y ruta como: **"C:\Microwin\Avc51\Intel.Hex".** Lo mismo ocurre si utiliza la opción ensamble Z80,

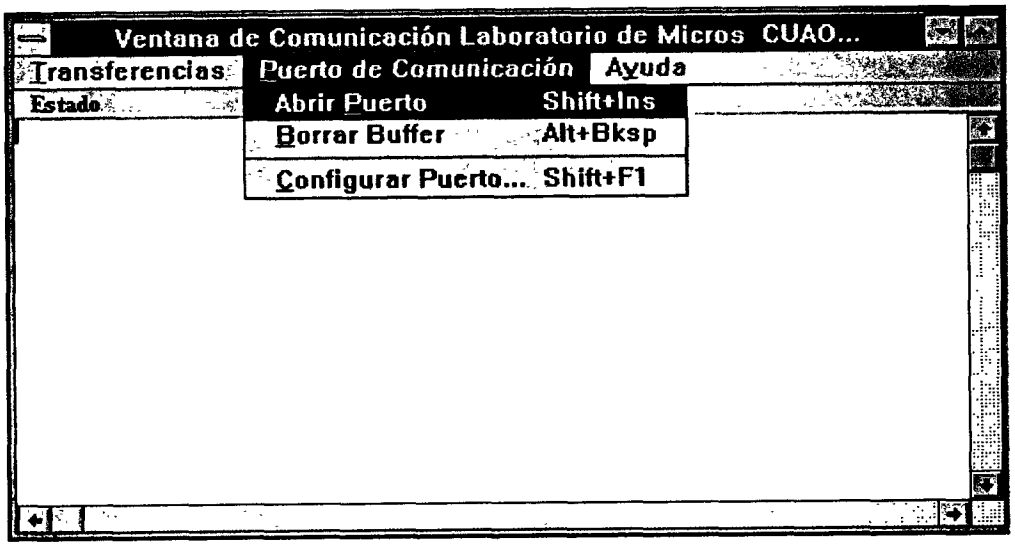
cuya ruta y nombre serán: **"C:\Microwin\Avcz80\Zilog.Hex"**

Salir

Termina la ejecución de la ventana de comunicaciones y regresa al Editor del **MICROWIN**.

El acceso inmediato a esta opción es por medio de la tecla **<F7>**

1.2.4.2. Comandos del Menú Puerto de Comunicaciones



Abrir puerto

Habilita a la venta de comunicaciones para recibir archivos de tipo texto y para

transmitir archivos tipo texto y hexadecimal. El acceso inmediato a esta opción es por medio de las teclas **<Shift>** + **<Ins>**

Nota: Si aparece el mensaje **"Puerto no disponible"** deberá elegir otro puerto diferente al **Com2,** en la opción configurar puerto. Recuerde que al seleccionar el puerto, debe elegir uno diferente al usado por el ratón

Borrar Búffer

Borra el contenido del búffer de desplazamiento de la ventana de comunicaciones, es decir, limpia la pantalla de la ventana de comunicaciones.

Las teclas abreviadas para este comando son **<Alt>** + **<Bksp>**.

Configurar Puerto

Este comando se debe ejecutar siempre que se desee hacer transferencias entre el Sistema de Desarrollo y MICROWIN, de tal manera que se configuren algunas de las características más importantes de la transmisión serial, como son la Rata de Baudios, Bits de Datos, Bits de Parada, Paridad, elección del puerto y control de flujo .

Para especificar los parámetros de comunicaciones:

1. En el menú Puerto de comunicaciones, elija Configurar puerto.
2. Elija las opciones correspondientes a su Sistema y al Sistema de Desarrollo que

utilice y luego el botón "Aceptar".

La figura 1.1. muestra la caja de diálogo que aparece cuando es accesado este comando. Las teclas de acceso directo son **<Shift> + <F1>**.

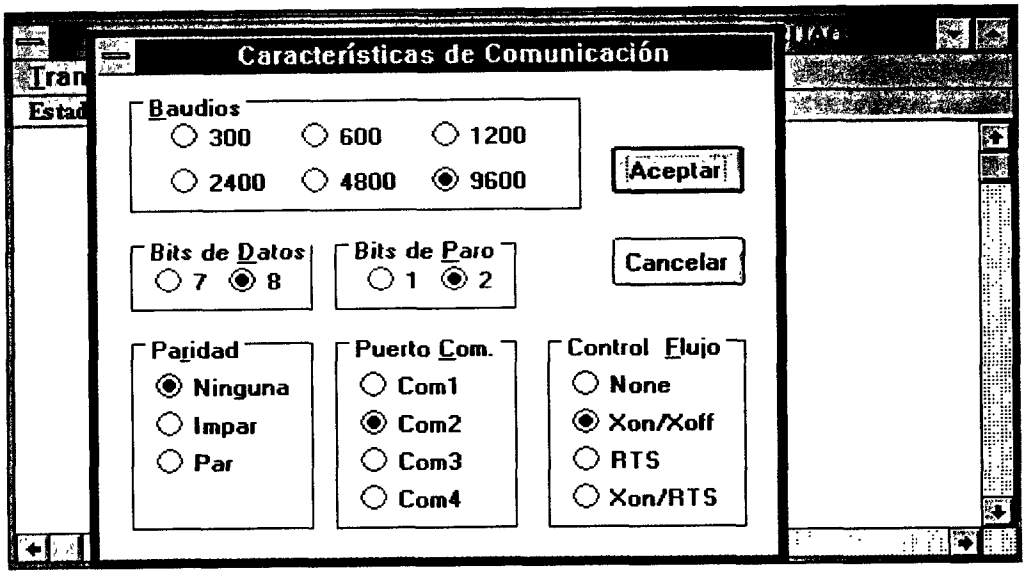


Figura 1

Para especificar los parámetros de comunicaciones:

1. En el menú Puerto de comunicaciones, elija **Configurar**.

2. Elija las opciones correspondientes a su sistema y al Sistema de Desarrollo que este utilizando y luego el botón "Aceptar".

Velocidad de transmisión en baudios: Especifica la velocidad de transmisión del Sistema de Desarrollo. Algunos sistemas de desarrollo pueden transmitir a diferentes "velocidades", dependiendo del cristal (Oscilador) que utilicen, por eso es conviente

que consulte el manual y elija una velocidad que puedan utilizar ambos sistemas.

Bits de datos: Especifica la cantidad de bits de información por cada paquete enviado del computador al Sistema de Desarrollo y viceversa.

Paridad: Especifica el tipo de paridad. Si seleccionó "8" bits de datos, elija **Ninguna** en esta sección.

Verificación de paridad: Determina el byte en el cual se ha detectado un error de paridad, de otra manera, verá un signo de interrogación (?) en el lugar donde el sistema de desarrollo encontró un error. Los signos de interrogación aparecerán en cada carácter que no haya sido transferido correctamente.

Puerto: Selecciona el puerto de comunicaciones que utilizará el Sistema de Desarrollo.

Bits de paro: Especifica el tiempo de espera entre caracteres transmitidos.

Control de flujo: Indica a la ventana de Comunicaciones de MICROWIN lo que debe hacer cuando el búfer esté demasiado lleno para recibir información desde el sistema remoto, elija **Ninguno** si el sistema de desarrollo si no utiliza algún método de control de flujo. Elija **XON/XOFF** si no sabe qué método se está utilizando.

1.2.5. Comandos del menú Ayuda

Este comando esta disponible tanto para el editor como para la ventana de comunicaciones. Ambos cuentan con los mismos comandos:

Contenido

Presenta el contenido general del sistema de Ayuda de **MICROWIN**.

Buscar ayuda sobre...

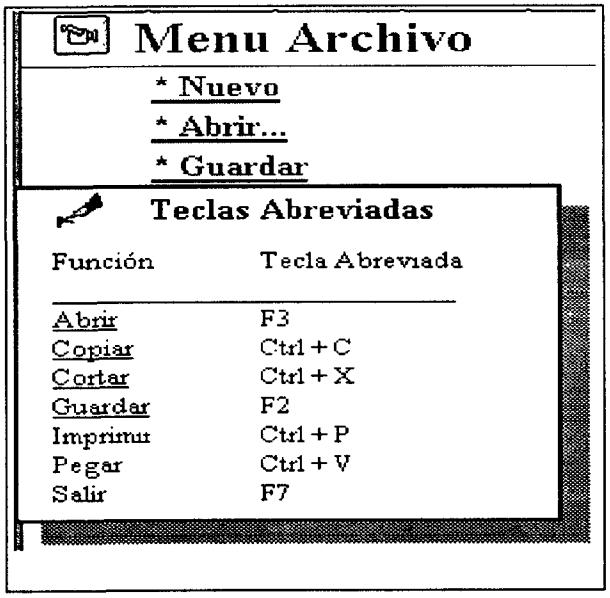
Busca en la Ayuda información acerca de temas sobre los que se desea más información.

Para utilizar el comando **Buscar ayuda sobre...**, escriba el tema o selecciónelo en la lista del cuadro de diálogo Buscar y, a continuación, elija el botón "Mostrar temas" para ver los temas relacionados. Para presentar un tema determinado de la lista de temas, selecciónelo y, después, elija el botón "Ir a".

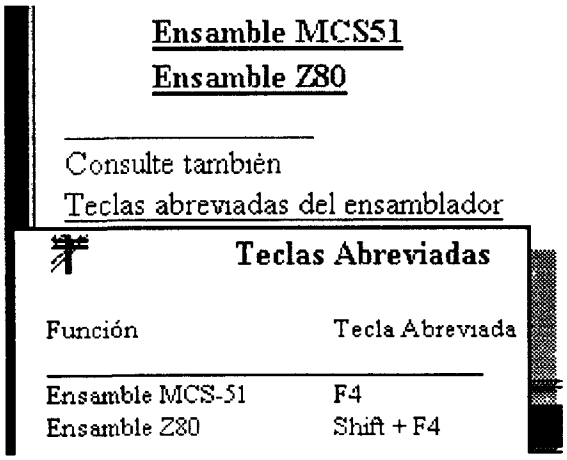
Para tener acceso directo al menú ayuda presione la tecla **<F1>.**

1.2.6 Teclas Abreviadas del software MICROWIN

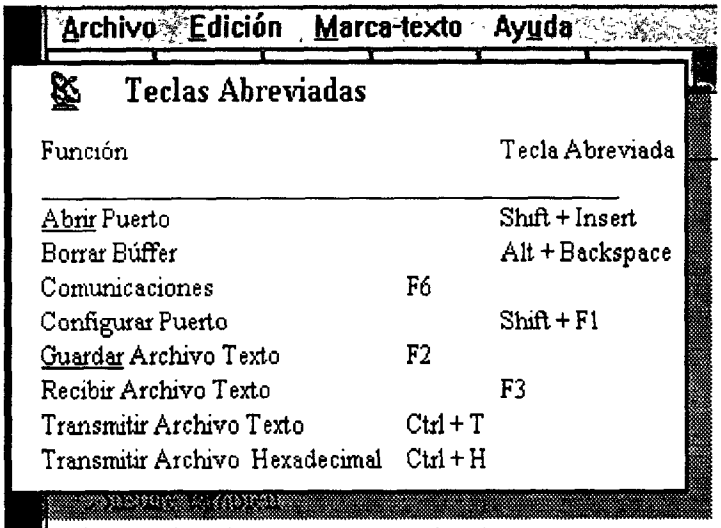
Teclas abreviadas del Editor



Teclas abreviadas del ensamblador



Teclas abreviadas del menú comunicaciones



Teclas abreviadas para comunicaciones